TAB 42

TAB 43

A COMBINED FORCE AND CUT ALGORITHM FOR HIERARCHICAL VLSI LAYOUT

G.J. WIPFLER, M. WIESEL, D.A. MLYNSKI

Institut für theoretische Elektrotechnik
Universität Karlsruhe, Germany

Abstract

This paper presents a new algorithm for the initial placement of hierarchical VLSI circuits. The components to be placed are orthogonal macrocells of variable shape and size. This algorithm combines the advantages of force directed placement and min-cut algorithm. It provides an initial placement which avoids overlapping between cells and includes an estimation of routting area. This algorithm is suitable for regular cell arrangements, too.

Introduction

This paper presents a procedure for the placement in the hierarchical design of integrated circuits. The components are macrocells of variable shape and size. Different approaches have been published. One class is based on force directed algorithms [1,2]. Another class are the min-cut algorithms [3,4,5].
In this paper the advantages of both classes are combined. The topological information of the circuit is calculated by a force directed placement algorithm. The cut algorithm uses the topology and the dimensions of the cells to calculate the final placement. The result is a placement algorithm for macrocells which also covers the more simple problem of regular cell arrangements.

Basic Placement Algorithms

Both classes of algorithms, the force directed and min-cut placement, will be shortly explained. They have been implemented to compare the different approaches using the same example.

Force Directed Placement

The force equations of the force directed placement programs [1,2] do neither consider the size nor the shape of the cells. A system of differential equations reflects the interconnection matrix and is solved by a modified Newton- Raphson method [2]. The advantage of this method is the fast calculation of the circuit topology. But in the solution cells may overlap because this method neglects the size and shape of the cells. Figure 1 shows the result of the force directed placement with relative positions, remaining forces and partial derivatives. The iteration of the force equations has been stopped at a given limit. Further iteration steps would not be significant for the topology. Figure 2 shows the overlaps between the cells for the placement given in Figure 1. Algorithms for the elimination of cell overlaps [2] have been proposed one for array carrier and one for non array carrier, respectively.

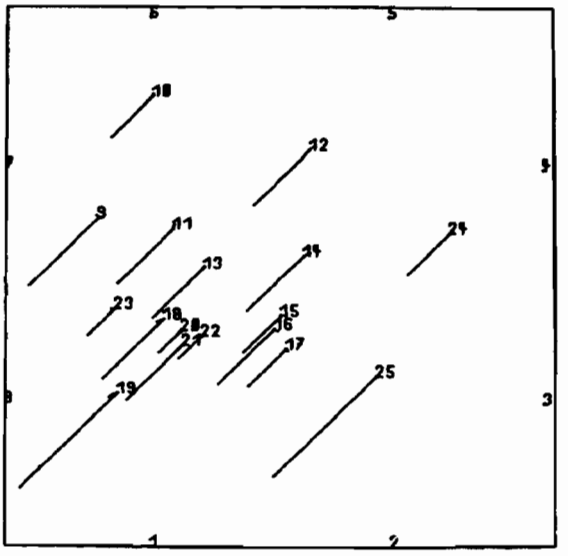


Figure 1 Relative positions, remaining forces, partial derivatives

19th Design Automation Conference

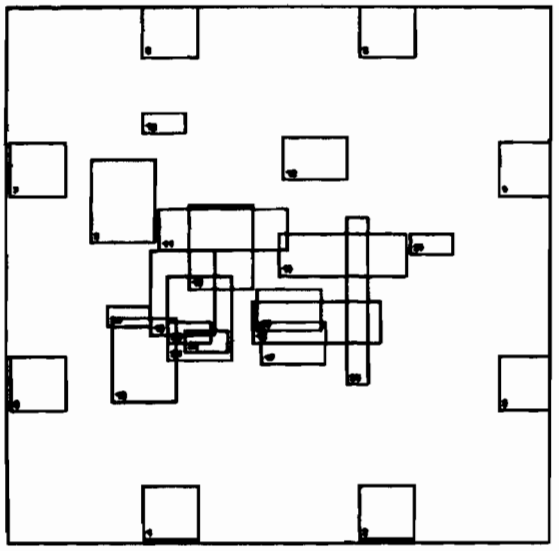


Figure 2 Force directed cell arrangement

Min-cut Placement

Different min-cut placement algorithms have been published [3,4,5]. The sequential block oriented approach has been implemented. Placement representation graphs as in [5] are not used.
Bipartitioning is done by the algorithm from [6,7]. The goal of this algorithm is to avoid a crowding of interconnections in the center of a circuit. This is achieved by minimizing the count of net-cuts in each partitioning step. The main advantage of this approach is the efficient consideration of the cell area during the block calculation.
Disadvantages are as follows: Due to its sequential nature, the min-cut procedure is a heuristic approach with local considerations. The results of the bipartitioning algorithm depend on the starting values. Moreover, the count of net-cuts is minimized locally only but not globally. This may be shown by the example in Figures 3 to 5. When bipartitioning graph G by the implemented min-cut procedure you get the four subsets A = {1,5}, B = {4,8}, C = {2,6} and D = {3,7} shown in Figure 4. The cut line S3' cuts 12 nets, the total number of cut nets is 26. Figure 5 shows a random partition of graph G cut by the lines S1", S2" and S3". The number of cut signals has decreased from 26 (Figure 4) to 22. Now the maximum number of cut nets cut by a single line (S1") is 10. This random partition of the graph G provides better results then the sequential partitioning that minimizes the count of net-cuts at each bipartitioning step. Parallel processing of more than one partitioning step is of high complexity and difficult to achieve. Additional constraints concerning the size of the blocks

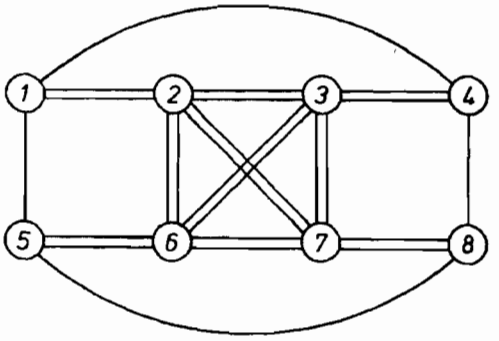


Figure 3 Graph G

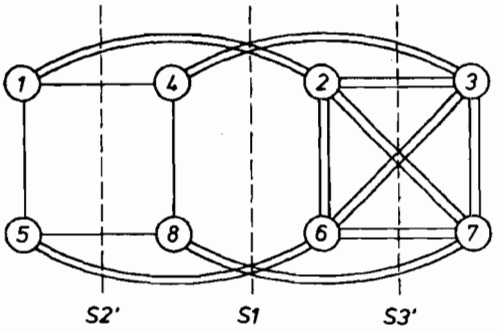


Figure 4 Partitioned graph G with all nets

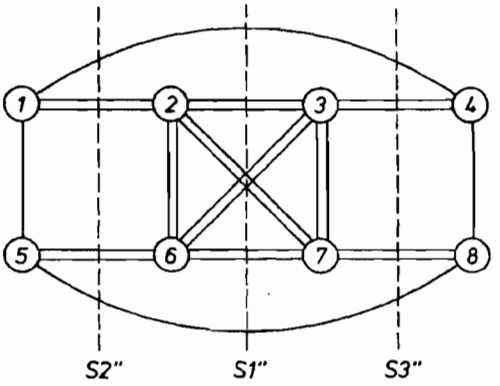


Figure 5 Random partitioning of graph G

(total area of all elements assigned to the block) are difficult to implement. The block calculation of the implemented algorithm considers the cell area, but not the cell shape. This causes overlaps of cells. Figure 6 and 7 show the the results of the min-cut algorithm for the same example as solved with the force directed placement algorithm (Figures 1 and 2).

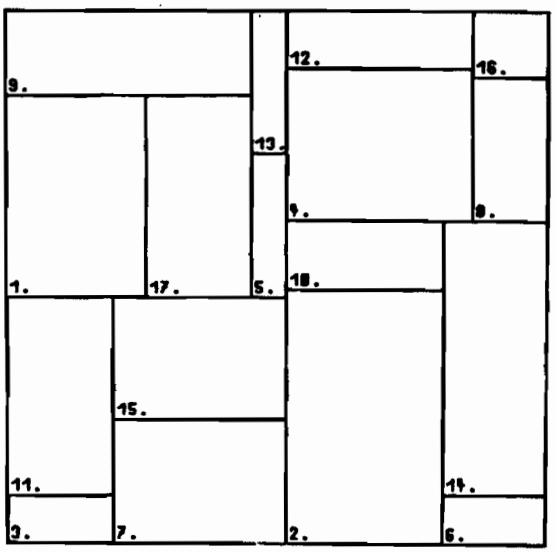


Figure 6 Blocks generated by min-cut

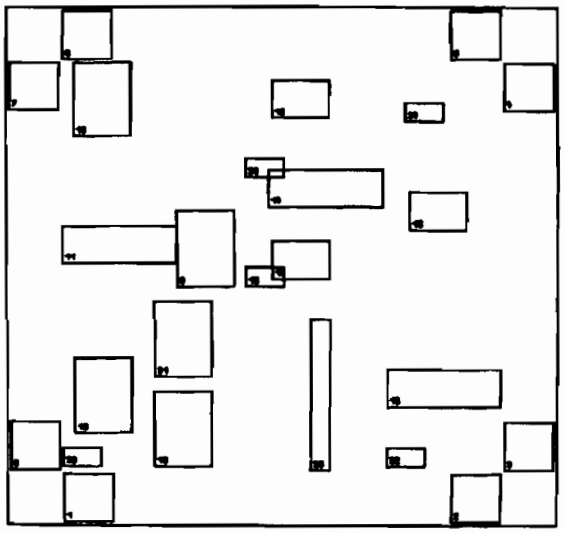


Figure 7 Min-cut cell arrangement

In the proposed Force and Cut Placement Algorithm (FOCUP) a new approach for elimination of overlaps shall be used as discussed in the following chapter. This method was used to get the min-cut placement without overlaps shown in Figure 8.

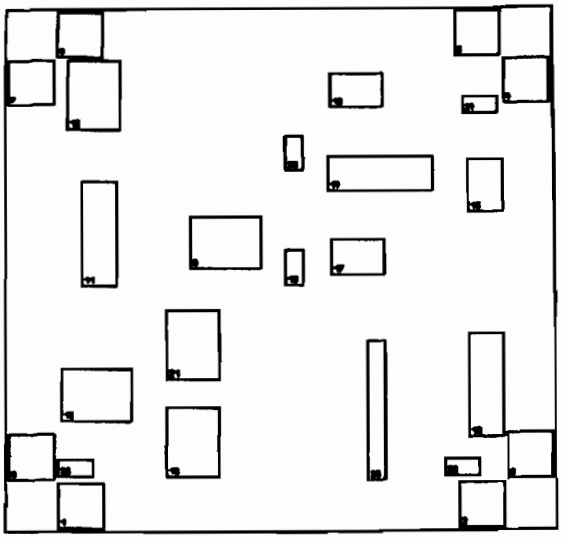


Figure 8 Min-cut solution

A New Force and Cut Placement Algorithm (FOCUP)

The basic idea of this new algorithm is to combine the advantages of both force directed and min-cut placement algorithms. The calculation of the placement is done in three steps:

1) A force directed placement algorithm calculates the relative cell positions (circuit topology).
2) A cut algorithm transforms this relative placement into a geometric cell arrangement considering cell area.
3) Considering cell shape, cells are moved and rotated in order to eliminate overlaps.

The last two steps have been extended to consider an estimated routing area. Several external constraints can be added.

Force Directed Algorithm

This is the implementation of the Phase I algorithm described in [2]. Some numerical constants have been changed, a different stopping criterion is used. Without considering the size and shape of cells the algorithm results in relative positions for all moveable cells (see Figure 1). The pads are at fixed locations.

Cut Algorithm

The cut algorithm used in step 2 does not need a bipartitioning, since topological information of the force directed algorithm is used for block division. The calculation of the cut-lines is done with respect to constraints such as size and shape of the cells, the number of cells per block or the minimization of critical signal nets. Different program options are available. Of course, these constraints influence the degree of the overlapping of the cells as well as the chip size and shape.
The block calculation is very simple. In case blocks of equal size are wanted the calculation is done as follows:

- Choose the cutdirection (horizontal or vertical).
- List the cells assigned to the actual block in an ascending order according to the y-positions (horizontal cut) or x-positions (vertical cut), respectively, as derived from the Force algorithm.
- Determine the number of cells in the predetermined order which fit in one half of the block area.
- Assign these cells to the left (bottom) new block, the remaining cells to the right (top) block.
- Compute the position of the cut line.

If there are two cells assigned to a block the direction of the cut-line is chosen in order to adapt the shape of the blocks to that of the cells. The block division ends if there is one cell assigned to each block. The result is shown by Figures 9 and 10 for the same example.

Overlap Elimination

After the calculation of the block divisions cells may overlap because the cells and blocks differ in shape. They can be moved and rotated in order to eliminate cell overlaps. The area of the blocks corresponds to that of the cells. Each block is extended in one (horizontal or vertical) direction to adapt the dimensions of the block to that of the cell. The cell may be rotated 90^0 to keep the extension as low as possible. To avoid new overlaps each block on the right (block extension in horizontal direction) or on the top (block extension in vertical direction) is moved. The shift of the blocks to be moved is that of the block to be extended. Each cell is moved with its block either in horizontal or vertical direction. Figures 11 and 12 demonstrate the overlap elimination.

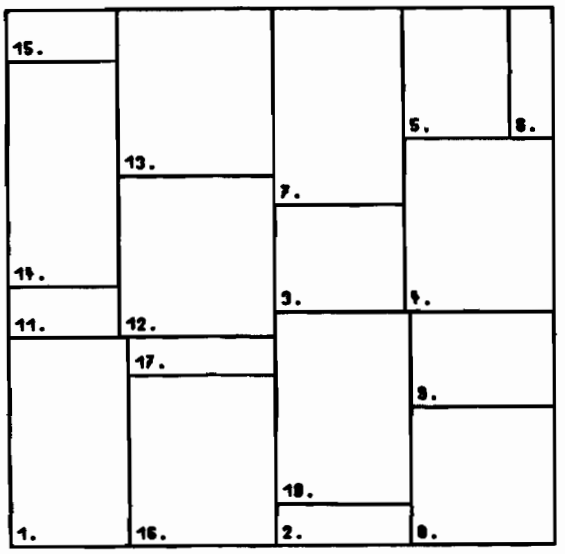


Figure 9 Blocks of the Cut procedure

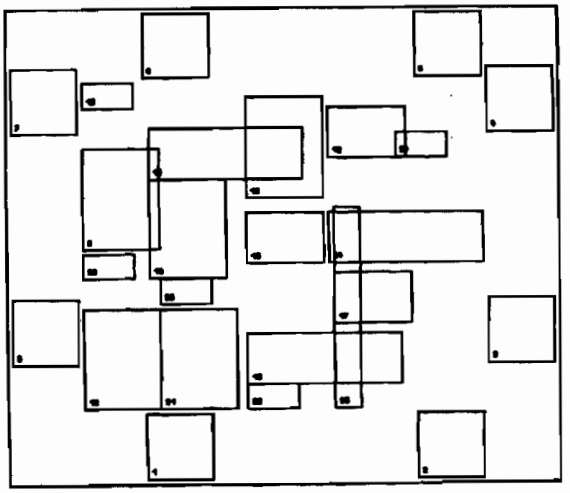


Figure 10 Cell arrangement according to Figure 9

Routing Area As An Additional Constraint

The new Force and Cut Algorithm (FOCUP) has been extended to handle estimated routing area. This is done by calculating an enclosing rectangle for each cell. The size of this rectangle is increased with respect to the number of terminals at each

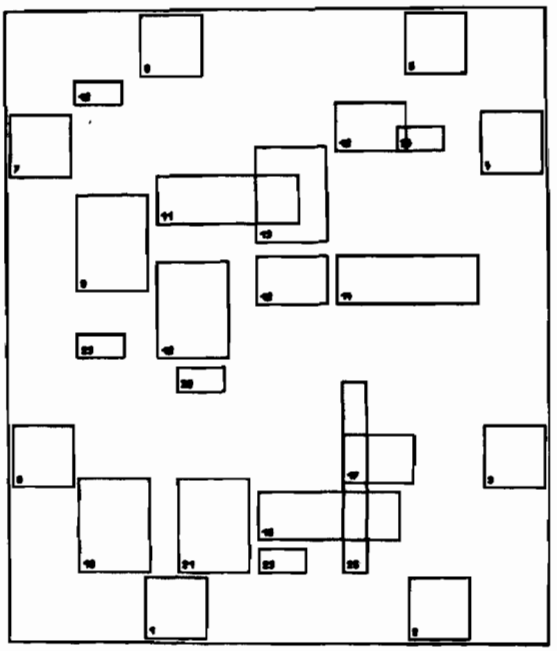


Figure 11 Result after moving step

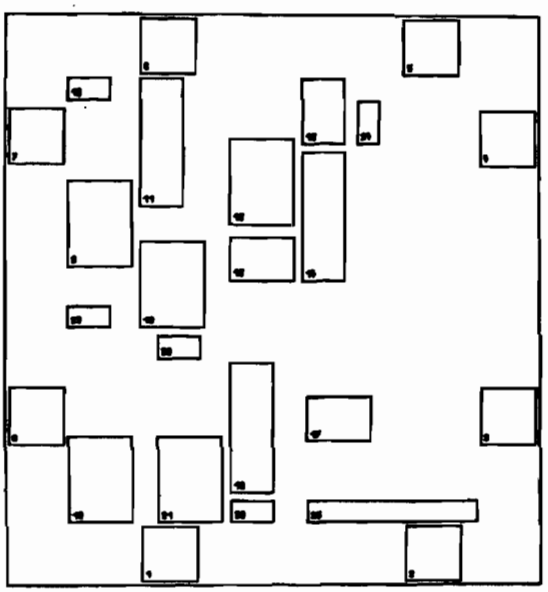


Figure 12 Result after rotating step

cell edge and a specified minimal distance between the cells. The algorithm can be extended to handle routing information in this step simultaneously.

Figure 13 shows the enclosing rectangles for two macrocells of different shape with different numbers of terminals.

During the global routing phase, a feedback to placement is necessary. Cells can

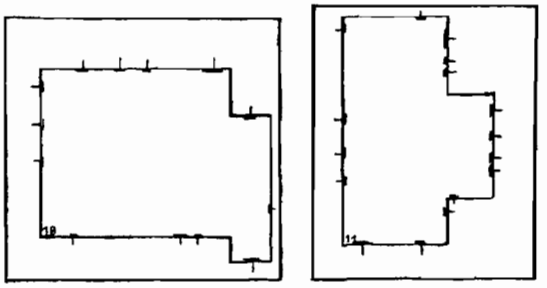


Figure 13 Macrocells and enclosing rectangles

be rotated and mirrored in order to improve the placement without placement recalculation (fast interactive mode) based on enclosing squares. The result of the FOCUP algorithm is shown for the same example in Figure 14. The estimation of the routing area was used as dicussed in this chapter.

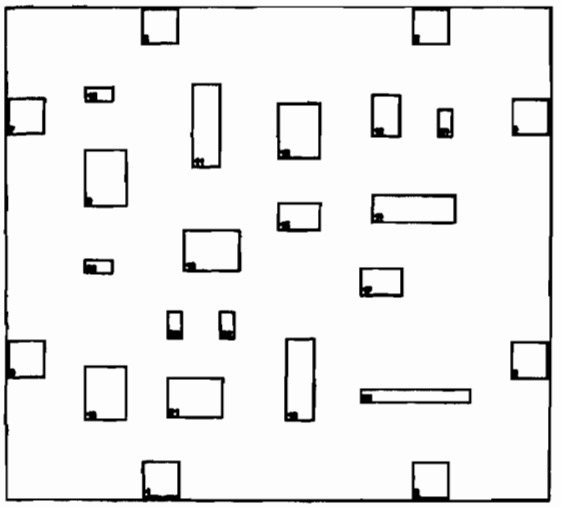


Figure 14 Final result of Force and Cut placement

Results

Optimized hand layouts are used to evaluate the new FOCUP programs. Figure 15 shows the manual placement with interconnections. This example was calculated with the Force and Cut Algorithm. Figure 16 shows the final placement of Figure 14 with interconnections routed automatically by the algorithm presented in [8].

When comparing the topology of the final placement, the results of the FOCUP algorithm (Figures 14 and 16) show a better matching with the hand layout (Figure 15)

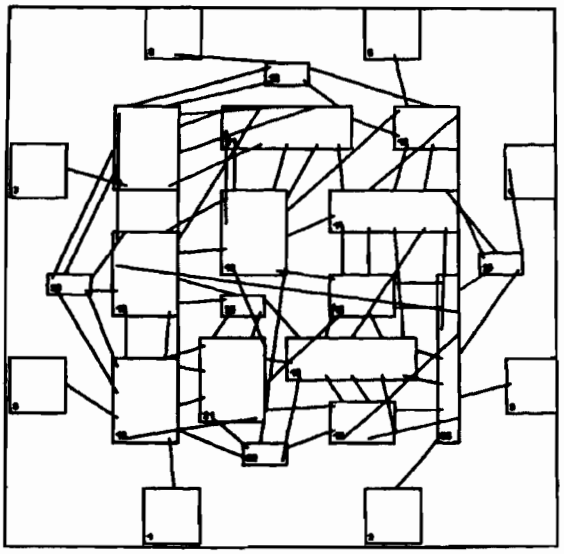


Figure 15 Manual constructed placement

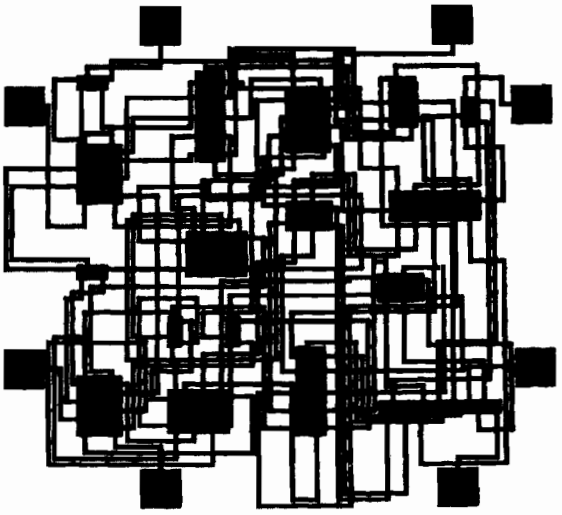


Figure 16 Final placement with interconnections

than those of the min-cut algorithm (Figure 9). Another important advantage of this approach is the short computation time. Normalized computation times for n = 25 cells are:

Min-cut	7
Force	1
Cut	1

The computational complexity of the min-cut algorithm is about $N=n^2 \log n$ [6]. The other two algorithms have the complexity of $N=n^2$, but the operational complexity (time requirement of a single step) of the Cut algorithm is considerably lower than that of the min-cut algorithm. The Force and Cut algorithm can be applied without modifications to regular structures like gate array or macrocells of very different size and shape.

Conclusions

The main advantages of the Force and Cut approach are the lower computation time. Apart from this different additional constraints and an estimation of the routing area can be handled. The algorithm is well suited for orthogonal building blocks of variable size and shape as well as gate arrays, masterslices or even printed circuit boards. It is fast enough for interactive placement optimization in combination with a global router.
All programs are in FORTRAN IV on a PDP 11/60 using a special data base system [8].

References

1 Quinn: "The Placement Problem As Viewed From The Physics Of Classical Mechanics", Proc. 12th Design Autom. Conf., pp 173-178, June 1975

2 Quinn,Breuer: "A Force Directed Component Placement Procedure For Printed Circuit Boards", IEEE Transactions on Circuits and Systems, Vol. CAS-26, June 1979

3 Breuer: "A Class Of Min-Cut Placement Algorithms", Proc. 14th Design Automation Conf., New Orleans, 1977 pp. 284-290

4 Breuer: "MIN-CUT Placement", J. Design Aut. and Fault-tolerant Computing, Vol. I, Nr. 4 Oct 1977, pp. 343-362

5 Lauther: "A MIN-CUT Algoritm For General Cell Assemblies Based On A Graph Representation", Proc. 16th Design Autom. Conf. 1979 pp. 1-10

6 Kernighan,Lin: "An Efficient Heuristic Procedure For Partitioning Graphs", Bell Syst. Techn. J., Vol. 49, Feb. 1970, pp. 291-308

7 Schweikert,Kernighan: "A Proper Model For Partitioning Of Electrical Circuits", Proc. Design Autom. Workshop, June 1972, pp. 56-62

8 H-J. Rothermel, M. Wiesel, D.A.Mlynski, "Routability Test With Variable Wire Width For Hierarchical Chip Design Based On A New Database System", ISCAS 1982, Rome, To be published

TAB 44

A NEW APPROACH FOR SOLVING THE PLACEMENT PROBLEM USING FORCE MODELS

Kurt J. Antreich, Frank M. Johannes and Fritz H. Kirsch

Technische Universitaet Muenchen, Lehrstuhl fuer Rechnergestuetztes Entwerfen
Munich, Federal Republic of Germany

ABSTRACT

In the computer-aided layout of electrical circuits, components can be placed using force models, which represent interconnection affinities as attracting and repelling forces. An improved approach is proposed to facilitate the placement task for printed circuit boards and for integrated circuits.

A new structuring of system equations is introduced, whose clarity affords much insight into the solution of the problem. Force equilibrium then determines the optimum relative positioning of components. A high degree of stability, a good numerical condition, and a low number of iteration steps in the quadratically-convergent solution method are among its important characteristics.

INTRODUCTION

The problem of laying out electrical circuits in printed-circuit-board or in integrated technology is too complex to be treated as a whole. It is thus divided into two independently solved problems, the placement and the wiring problems [1].

In holding computing costs within reasonable limits, placement cannot be performed taking into account all geometrical and electrical restrictions of the wiring problem. Rather, an attempt is made during the placement procedure to account for all wiring restrictions at once by means of a weighting function, and then to minimize this function by proper component arrangement.

Force models have previously been used in solving the placement problem [1,2,3,4,5]. Here, the components are represented by mass-free bodies, and the interconnection affinities by couples of forces active between body pairs. These methods differ in the applied model variations as well as in the various solution strategies. One generally attempts to improve upon an initial placement by interchanging members of component pairs. Using a force model, pairs can be selected which are particularly well-suited for interchanging.

A recent paper [6] describes a force model utilizing a nonlinear system of equations. When this system is solved, interconnected components are optimally assigned to adjoining locations. Among various solution methods discussed in [6], a one-at-a-time procedure is described in detail.

Using the model in [6] as an example, we introduce a problem-oriented structuring of system equations, which corresponds largely to the structure of system equations for electrical circuit analysis. By adopting these measures, the nonlinear relationships occuring in the complete description of the problem can be isolated in independent equations. The solution equations for determining force equilibrium in the plane are formulated using complex variables to describe the system parameters. The concepts introduced in the following chapters are illustrated in Appendix A.

PROBLEM FORMULATION

The circuit topology can be described by an interconnection set N of l-place component relations:

$$N = (N_2, N_3, \ldots, N_l, \ldots) \tag{1}$$

Each l-place relation N_l is a set of component-l-tuples. Each of these l-tuples represents an interconnection net of equipotential with l components.

We treat the placement problem as an l-place assignment problem [7,8,9]:

> Given a set of components and a set of locations, assign all components to locations, such that the sum of all interconnection lengths (measured as minimum spanning trees, e.g.) is minimized.

To simplify this l-place problem, one can change it into a quadratic one [2] by replacing each l-place relation N_l by a binary relation N_{l2} such that all possible $l(l-1)/2$ component pairs are formed from each l-tuple in N_l. Each interconnection net with l components is thus represented by a complete binary graph with $l(l-1)/2$ edges.

The interconnection set N thus becomes a set $\hat{N}_b$ with only binary relations N_{l2}. $\hat{N}_b$ can therefore be substituted by a single binary relation N_b on the component set M and represented by a binary graph.

$$N \rightarrow \hat{N}_b = (N_{22} = N_2, N_{32}, \ldots, N_{l2}, \ldots) \tag{2}$$

$$\hat{N}_b \rightarrow N_b = \{(i,j) \in M \times M \mid \exists_{l \in (2,3,\ldots)} (i,j) \in N_{l2}\} \tag{3}$$

According to the above substitution, each interconnection net σ receives a weight $l(l-1)/2$, depending on the number of edges it has. However, only l-1 edges are needed to realize an interconnection net as a tree. To obtain the best adaption of the quadratic assignment problem to the original placement problem, one assigns a weight of l-1 to each interconnection net. This can obviously be achieved by giving each of the $l(l-1)/2$ edges the weight of $k(i,j) = L_\sigma^{(i,j)} = 2/l$ [2,10]. If a component pair (i,j) in N_b has more than one net σ in common, the corresponding sum of the weights is used:

$$k_{(i,j)} = \sum_\sigma L_\sigma^{(i,j)} \tag{4}$$

$$\text{where } L_\sigma^{(i,j)} = \begin{cases} 2/l & \text{if } (i,j) \rightarrow \text{l-tupel } \sigma \\ 0 & \text{otherwise} \end{cases}$$

The weights of all connected component pairs in N_b can be expressed as a diagonal matrix $\underset{\sim}{K}_{cc}$:

$$\underset{\sim}{K}_{cc} = \text{diag}(\ldots, k_{(i,j)}, \ldots) \quad \text{for all } (i,j) \in N_b \tag{5}$$

CH1681-6/82/0000-0481 $00.75 © 1982 IEEE

STRUCTURING OF THE PROBLEM

The solution of the problem is further simplified by changing the discrete combinatorial quadratic assignment problem into a continuous dimensioning problem. One generally uses a force model (see introduction), in which the components are represented by mass-free bodies and where the weights of component pairs are modelled by pairs of forces between bodies.

To obtain a well-structured problem formulation, we advise the application of the incidence matrix $\underset{\sim}{A}$, which completely describes the relation N_b and its representation as a binary graph. To the rows of $\underset{\sim}{A}$ components or vertices are assigned, letting m be the number of movable components and n the number of nonmovable ones. To the columns of $\underset{\sim}{A}$ component pairs or edges are assigned, letting c be the number of connected component pairs and d the number of disconnected ones. This yields a partitioning of the incidence matrix $\underset{\sim}{A}$:

$$\underset{\sim}{A} = \begin{array}{c} \overbrace{}^{c} \; \overbrace{}^{d} \\ \left[\begin{array}{c|c} \underset{\sim}{A}_{mc} & \underset{\sim}{A}_{md} \\ \hline \underset{\sim}{A}_{nc} & \underset{\sim}{A}_{nd} \end{array}\right] \end{array} \begin{array}{l} \left.\right\} m \\ \left.\right\} n \end{array} \begin{array}{l} \longleftarrow \text{component pairs (edges)} \\ \downarrow \text{components (vertices)} \end{array} \qquad (6)$$

In each column of $\underset{\sim}{A}$, the matrix element +1 denotes the vertex i, and the element -1 the vertex j of a given connected component pair (i,j). The edges are thus given arbitrary reference directions used for distance and force vectors between component pairs. For disconnected component pairs, the complementary set of the graph's edges is to be applied. In this model no forces are acting between fixed and movable components of disconnected pairs, i.e., $\underset{\sim}{A}_{nd}=0$.

$\underline{z}=\begin{bmatrix}\underline{z}_m\\ \underline{z}_n\end{bmatrix}$	actual locations of	m movable / n fixed	components
$\underline{f}=\begin{bmatrix}\underline{f}_m\\ \underline{f}_n\end{bmatrix}$	resulting forces on	m movable / n fixed	components
$\underline{\gamma}=\begin{bmatrix}\underline{\gamma}_c\\ \underline{\gamma}_d\end{bmatrix}$	distances between	c connected / d disconnected	component pairs
$\underline{w}=\begin{bmatrix}\underline{w}_c\\ \underline{w}_d\end{bmatrix}$	distance-oriented forces between	c connected / d disconnected	component pairs

Table 1: Parameter vectors

To describe the plane force model, we suggest the use of complex variables, letting $z=x+jy$ stand for the actual component locations, $f=f_x+jf_y$ for the resulting forces working on the components, $\gamma=\alpha+j\beta$ for the distances between component pairs, and $w=u+jv$ for the distance-oriented forces. In Table 1 these complex vectors are listed and partitioned according to the structure of the incidence matrix.

In structuring the problem description, one can differentiate between three basic types of equation:

a) The force equations FE, which describe by means of the incidence matrix $\underset{\sim}{A}$ the linear relationship between the distance-oriented forces $\underline{w}$ and the resulting forces $\underline{f}$ working on the components

$$\text{FE:} \quad \begin{bmatrix}\underline{f}_m\\ \underline{f}_n\end{bmatrix} = \begin{bmatrix}\underset{\sim}{A}_{mc} & \underset{\sim}{A}_{md}\\ \underset{\sim}{A}_{nc} & \underset{\sim}{A}_{nd}\end{bmatrix} \cdot \begin{bmatrix}\underline{w}_c\\ \underline{w}_d\end{bmatrix} \qquad (7)$$

b) The spring equations SE, which describe the relationship between the distances $\underline{\gamma}$ and the distance-oriented forces $\underline{w}$ by means of the diagonal matrices $\underset{\sim}{K}_{cc}$ (see (5)) and $\underset{\sim}{K}_{dd}$

$$\text{SE:} \quad \begin{bmatrix}\underline{w}_c\\ \underline{w}_d\end{bmatrix} = \begin{bmatrix}-\underset{\sim}{K}_{cc} & \underset{\sim}{0}\\ \underset{\sim}{0} & \underset{\sim}{K}_{dd}\end{bmatrix} \cdot \begin{bmatrix}\underline{\gamma}_c\\ \underline{\gamma}_d\end{bmatrix} \qquad (8)$$

c) The distance equations DE, which describe the linear relationship between the actual component locations $\underline{z}$ and the distances $\underline{\gamma}$ of component pairs by means of the transposed matrix $\underset{\sim}{A}^T$

$$\text{DE:} \quad \begin{bmatrix}\underline{\gamma}_c\\ \underline{\gamma}_d\end{bmatrix} = \begin{bmatrix}\underset{\sim}{A}_{mc}^T & \underset{\sim}{A}_{nc}^T\\ \underset{\sim}{A}_{md}^T & \underset{\sim}{A}_{nd}^T\end{bmatrix} \cdot \begin{bmatrix}\underline{z}_m\\ \underline{z}_n\end{bmatrix} \qquad (9)$$

In the force model [6], the resulting forces $\underline{f}_m$ working on the m movable components are composed of three parts. The first part $\underline{f}_{m1}$ results from forces of attraction between connected component pairs. The second part $\underline{f}_{m2}$ results from forces of repulsion between movable, disconnected component pairs. A third part $\underline{f}_{m3}$ serves to compensate the forces of the fixed components working on the entirety of the movable components by means of a force acting equally on each movable component.

$$\begin{aligned} &\text{FE:} && \underline{f}_{m1} = \underset{\sim}{A}_{mc} \cdot \underline{w}_c \\ &\text{SE}_{(\text{linear})}\text{:} && \underline{w}_c = -\underset{\sim}{K}_{cc} \cdot \underline{\gamma}_c \\ &\text{DE:} && \underline{\gamma}_c = \underset{\sim}{A}_{mc}^T \cdot \underline{z}_m + \underset{\sim}{A}_{nc}^T \cdot \underline{z}_n \\ & && \underline{f}_{m1} = -\underset{\sim}{A}_{mc} \cdot \underset{\sim}{K}_{cc} \cdot \underset{\sim}{A}_{mc}^T \cdot \underline{z}_m - \underset{\sim}{A}_{mc} \cdot \underset{\sim}{K}_{cc} \cdot \underset{\sim}{A}_{nc}^T \cdot \underline{z}_n \\ & && \underline{f}_{m1} = -\underset{\sim}{B}_{mm} \cdot \underline{z}_m - \underline{b}_m \end{aligned}$$

Table 2: First part of resulting forces produced by attracting forces.

$$\begin{aligned} &\text{FE:} && \underline{f}_{m2} = \underset{\sim}{A}_{md} \cdot \underline{w}_d \\ &\text{SE}_{(\text{nonlinear})}\text{:} && \underline{w}_d = \underset{\sim}{K}_{dd}(\underline{\gamma}_d) \cdot \underline{\gamma}_d \\ &\text{DE:} && \underline{\gamma}_d = \underset{\sim}{A}_{md}^T \cdot \underline{z}_m \\ & && \underline{f}_{m2} = \underset{\sim}{A}_{md} \cdot \underset{\sim}{K}_{dd}(\underline{\gamma}_d) \cdot \underset{\sim}{A}_{md}^T \cdot \underline{z}_m \\ & && \underline{f}_{m2} = \underset{\sim}{R}_{mm}(\underline{\gamma}_d) \cdot \underline{z}_m \end{aligned}$$

Table 3: Second part of resulting forces produced by repelling forces.

Tables 2 and 3 describe how to calculate the parts $\underline{f}_{m1}$ and $\underline{f}_{m2}$ of the resulting forces from the basic equations (7), (8) and (9). $\underline{f}_{m1}$ and $\underline{f}_{m2}$, working on the movable components, are dependent on the locations $\underline{z}_m$ of the actual movable components.

Looking at the forces of attraction between all connected component pairs, one sees that the distance-oriented forces are proportional to the respective distances (Hooke's law). Thus, each individual SE of Table 2 has the principally linear form

$$\text{SEc:} \quad w = -k \cdot \gamma \quad . \qquad (10)$$

taking k as the positive real spring constant or edge weight (see (5)).

Examining the forces of repulsion reveals that the distance-oriented forces are obtained independently of the distances $\underline{\gamma}$ between the respective

component pairs [6]. Each individual SE of Table 3 is thus given the principally nonlinear form

$$\text{SEd:} \quad w = \frac{k_0}{|\gamma|}\cdot\gamma \quad . \tag{11}$$

In [6] k_0 was given a arbitrary constant value, such that the sum of all spring constants between attracted pairs of movable components is equal to $d\cdot k_0$. If the index κ denotes all movable, disconnected component pairs,

$$\underset{\sim}{K}_{dd} = \text{diag}(\ldots,\frac{k_0}{|\gamma_\kappa|},\ldots) \quad . \tag{12}$$

In calculating the third part $\underline{f}_{m3}$ of the resulting forces on movable components, the total force acting between movable and fixed components is needed. This force must be compensated, since in a condition of equilibrium the movable components would otherwise be placed around the fixed components (e.g., around a connector).

The resulting forces $\underline{f}_{n1}$ working on the fixed components can be calculated analoguously to those shown in Table 2.

$$\underline{f}_{n1} = -\underset{\sim}{A}_{nc}\cdot\underset{\sim}{K}_{cc}\cdot\underset{\sim}{A}^T_{mc}\cdot\underline{z}_m - \underset{\sim}{A}_{nc}\cdot\underset{\sim}{K}_{cc}\cdot\underset{\sim}{A}^T_{nc}\cdot\underline{z}_n \tag{13}$$

The sum S of vector components from $\underline{f}_{n1}$ clearly represents the total force.

$$S(\underline{f}_{n1}) = \underline{1}^T_n\cdot\underline{f}_{n1} \tag{14}$$

The sum of all elements of $\underline{f}_{n1}$ can be determined simply using the inner product with the vector $\underline{1}_n$, which contains n ones as elements.

To compensate for the total force between fixed and movable components, the m-th part of this force is distributed evenly among all movable components.

$$\underline{f}_{m3} = \frac{1}{m}\cdot\underline{1}_m\cdot\underline{1}^T_n\cdot\underline{f}_{n1} \tag{15}$$

As forces of repulsion are present in the force model only between movable components,

$$\underline{1}^T_m\cdot\underline{f}_{m2} = 0 \tag{16}$$

and

$$\underline{1}^T_n\cdot\underline{f}_{n1} = -\underline{1}^T_m\cdot\underline{f}_{m1} \quad . \tag{17}$$

are valid. Using (17) and Table 2, (15) yields

$$\underline{f}_{m3} = -\frac{1}{m}\cdot\underline{1}_m\cdot\underline{1}^T_m\cdot\underline{f}_{m1} = \frac{1}{m}\cdot\underline{1}_m\cdot\underline{1}^T_m\cdot(\underset{\sim}{B}_{mm}\cdot\underline{z}_m+\underline{b}_m) \tag{18}$$

$$\underline{f}_{m3} = \underset{\sim}{H}_{mm}\cdot\underline{z}_m+\underline{h}_m \quad . \tag{19}$$

$\underset{\sim}{H}_{mm}$ consists of m identical rows, composed from the element sums of the columns in $\underset{\sim}{B}_{mm}$, which are weighted with the value 1/m. The calculation of $\underline{h}_m$ from $\underline{b}_m$ is correspondingly straight forward.

SYSTEM EQUATIONS FOR THE FORCE MODEL

From Tables 2 and 3 and (19), the m resulting forces on the m movable components can be derived as a function of the m actual component locations.

$$\begin{aligned}\underline{f}_m = \underline{f}_{m1}+\underline{f}_{m2}+\underline{f}_{m3} = &(-\underset{\sim}{B}_{mm}+\underset{\sim}{H}_{mm})\cdot\underline{z}_m \quad \text{(linear)}\\ &+\underset{\sim}{R}_{mm}(\underline{\gamma}_d)\cdot\underline{z}_m \quad \text{(nonlinear)}\\ &-\underline{b}_m+\underline{h}_m \quad \text{(constant)}\end{aligned} \tag{20}$$

From Tables 2 and 3, one can see how to construct the matrices $\underset{\sim}{B}_{mm}$ and $\underset{\sim}{R}_{mm}$ of the system (20). The information about N or N_b, $\underset{\sim}{K}_{cc}$ and k_0 (see (1),(3),(5),(12)) is stored in lists. Thus, the individual matrix elements of $\underset{\sim}{B}_{mm}$ and $\underset{\sim}{R}_{mm}$ can be obtained without performing the matrix multiplications of the sparse incidence matrix $\underset{\sim}{A}$.

The analogy to the system equations of electrical circuit analysis was fundamental to the structuring of the system equations for the force model. The FE correspond to Kirchhoff's current law, the SE to Ohm's law for one-port circuits, and the DE to Kirchhoff's voltage law. The construction of matrices $\underset{\sim}{B}_{mm}$ and $\underset{\sim}{R}_{mm}$ corresponds largely to the nodal-admittance matrix, which is well known in connection with nodal voltage analysis.

In formulating an approximate solution to the placement problem, the following equilibrium equations for the force model are used:

$$\underline{f}_m \overset{!}{=} \underline{0} \tag{21}$$

Thus, an optimum relative component positioning $\underline{z}_m$ can be attained by solving this nonlinear system of equations. The movable components may then be assigned to their final locations interactively: First, some suitable movable components are placed in proper slots, thus increasing the number of fixed components. Where needed, the optimum relative placement for the remaining movable components can in turn be determined from the reduced system equations. The procedure is continued until each component has been assigned to a slot.

In solving the equation system, one must bear in mind that the sum of the resulting forces working on the movable components is equal to zero:

$$\underline{1}^T_m\cdot\underline{f}_m = \underline{1}^T_m\cdot(\underline{f}_{m1}+\underline{f}_{m2}+\underline{f}_{m3}) = 0 \quad . \tag{22}$$

Taking into account (16), as well as

$$\underline{1}^T_m\cdot\underline{f}_{m3} = -\frac{1}{m}\cdot\underline{1}^T_m\cdot\underline{1}_m\cdot\underline{1}^T_m\cdot\underline{f}_{m1} = -\frac{1}{m}\cdot m\cdot\underline{1}^T_m\cdot\underline{f}_{m1} \quad . \tag{23}$$

resulting from (15) and (17), the validity of (22) becomes apparent. The system (21) thus contains one linearly dependent equation. This results in a degree of freedom in determining the equilibrium. Thus, the location of any movable component, e.g., can be arbitrarily specified. In this context, we suggest fixing the center of gravity s of all movable components on the geometric center of the placement plane. In system (21), an arbitrary equation is replaced by the gravity equation:

$$\frac{1}{m}\cdot\underline{1}^T_m\cdot\underline{z}_m = s \tag{24}$$

If the center of gravity is placed at the origin of the complex placement plane (s=0+j0), a particularly good numerical condition can be obtained for the solution of the system equations.

The model can be further improved if the constant k_0 of (12) which determines the magnitude of the forces of repulsion, is treated as an additional independent variable. Thus, one more real equation is needed to solve the system of equations. To adjust the area of the solution $\underline{z}_m$ to the shape of the placement plane, we suggest keeping a norm of the locations $\underline{z}_m$ constant [12]:

$$\|\underline{z}_m\|^2 \approx \underline{z}^H_m\cdot\underline{z}_0+\underline{z}^H_0\cdot\underline{z}_m-\underline{z}^H_0\cdot\underline{z}_0 \overset{!}{=} \text{const} \tag{25}$$

SOLUTION EQUATIONS FOR FORCE EQUILIBRIUM

One applicable method for solving the nonlinear equation system (21) is fixed-point iteration

$$\underline{z}^{(\nu+1)}_m = (\underset{\sim}{B}_{mm}-\underset{\sim}{H}_{mm})^{-1}\cdot[\underset{\sim}{R}_{mm}(\underline{\gamma}^{(\nu)}_d)\cdot\underline{z}^{(\nu)}_m-\underline{b}_m+\underline{h}_m] \tag{26}$$

This procedure is linearly convergent and does not require differentiation or linearization [12].

In this paper, we apply the multidimensional, quadratically convergent Newton-Raphson iteration method for solving the nonlinear equation system (21). Starting with an initial placement $\underline{z}_m^{(0)}$, a rule for computing an improved solution $\underline{z}_m^{(0)}$ can be attained by linearizing (21). This process is repeated using the improved placement until a solution $\underline{z}_m^{(\mu)}$ with sufficient numerical precision is reached.

The advantages in structuring the problem as presented here become particularly obvious when determining the linearized form of (21). Its nonlinearity is isolated in the SE describing forces of repulsion, while all other equations remain linear.

The linear form of the solution equations for the $(\mu+1)$th iteration step yields (see Appendix B or [11])

$$\hat{\underline{f}}_m^{(\mu+1)} = \underline{0} = (-\underset{\sim}{B}_{mm} + \underset{\sim}{H}_{mm} + \underset{\sim}{P}_{mm}^{(\mu)}) \cdot \underline{z}_m^{(\mu+1)} - \underset{\sim}{Q}_{mm}^{(\mu)} \cdot \underline{z}_m^{*(\mu+1)} - \underline{b}_m + \underline{h}_m + \underline{p}_m^{(\mu)} \quad (27)$$

The difference from the system equations (21) is clear. As explained above an arbitrary equation from system (27) is replaced by the gravity equation (24). The matrices $\underset{\sim}{B}_{mm}$, $\underset{\sim}{H}_{mm}$, and the vectors $\underline{b}_m$, $\underline{h}_m$ are computed before starting the iterative process. Beginning iteration with an initial placement $\underline{z}_m^{(0)}$, the distances $\gamma^{(0)}$ are calculated using the DE. Using (40), (41), (42) and (43) in Appendix B, the real matrix $\underset{\sim}{P}_{mm}^{(0)}$, the complex matrix $\underset{\sim}{Q}_{mm}^{(0)}$ and the complex vector $\underline{p}_m^{(0)}$ are then determined. After these preliminary calculations, the system (27) yields an improved solution in the form of a real part $\underline{x}_m^{(1)}$ and an imaginary part $\underline{y}_m^{(1)}$ of the complex vector $\underline{z}_m^{(1)}$. The iterative process is then continued using the improved solution $\underline{z}_m^{(1)}$.

The decreasing norm of the resulting force vector $\underline{f}_m$ is used as an assessment criterion for process stability:

$$\| \underline{f}_m^{(\mu+1)} \| \overset{!}{<} \| \underline{f}_m^{(\mu)} \| \quad \text{where} \quad \| \underline{f}_m \| = \sqrt{\underline{f}_m^{*T} \cdot \underline{f}_m} \quad (28)$$

In cases where instability in the iterative process might arise, the relative linearization error of the distance-oriented forces $\underline{w}_d$ can be limited by

$$\frac{\| \hat{\underline{w}}_d - \underline{w}_d \|}{\| \underline{w}_d \|} \leq \delta_w \quad (29)$$

which reduces the relative change of the locations:

$$\frac{\| \underline{z}_m^{(\mu+1)} - \underline{z}_m^{(\mu)} \|}{\| \underline{z}_m^{(\mu)} \|} = \varepsilon^{(\mu+1)} \quad (30)$$

Where location changes are small, one can halt the iterative process with reliable termination criteria due to its quadratic convergence:

$$\varepsilon^{(\mu+1)} \sim \varepsilon^{(\mu)2} \quad (31)$$

RESULTS

The proposed placement method was applied to several circuits, some of whose optimum placement was already known. From various randomly-generated initial placements for the movable components, the optimum final positioning was attained in all cases in 2 to 5 iteration steps.

The test cases were calculated using a CYBER 175 computer. In Appendix C, 25 movable components were placed using 3 iteration steps. Each step required roughly 0.3 seconds of computation time. The placement of 100 movable components of another circuit was computed in 3 iteration steps of 7 seconds each.

Insights gained thus far in solving the problem have shown that the application of the Newton-Raphson method to the problem at hand is characterized by a high degree of stability. Starting with an arbitrarily-chosen initial placement, a 'unique' final placement was obtained in all cases in a low number of iteration steps. Surprisingly, the stability was virtually unimpaired by large linearization errors in the initial iteration steps. The size of the system, i.e. the number of movable components, is limited only by computation time. Very large systems can be solved, due, in our opinion, to the good numerical condition of the problem when properly normalized.

The clarity of the problem structuring with respect to well-known formulations of the system equations makes it possible, on the one hand, to produce technically useful solutions as program modules, and, on the other hand, to pursue investigation of methods leading to new variations of the model and to cost-reducing solution strategies.

REFERENCES

[1] Soukup, J., Circuit layout, Proceedings IEEE, vol.69 [1981], 1281 - 1304.

[2] Hanan, M., Wolff, P.K. and Agule, B.J., Some experimental results on placement techniques. Proceedings 13th Design Automation Conference, June 1976, 214 - 224.

[3] Fisk, C.J., Caskey, D.L. and West, L.E., ACCEL: Automated circuit card etching layout, Proc. IEEE, vol.55 [1967], 1971 - 1982.

[4] Scanlon, F.T., Automated placement of multi-terminal components. Proc. 8th Design Automation Workshop, July 1971, 143 - 154.

[5] Shupe, C.F., Automatic component placement in the NOMAD system. Proceedings 12th Design Automation Conference, June 1975, 162 - 172.

[6] Quinn, N.R. and Breuer, M.A., A force directed component placement procedure for printed circuit boards. Transactions IEEE CAS-26 [1976], 377 - 388.

[7] Engl, W.L., Mlynski, D.A. and Parnards, P., Theory of multiplace graphs. Transactions IEEE CAS-22 [1975], 759 - 767.

[8] Koller, K. and Lauther, U., Computer-aided layout of large-scale integrated circuits based on cells (in German). Nachrichtentechnische Zeitschrift 31 [1978], 906 - 910.

[9] Goto, S. and Kuh, E.S., An approach to the two-dimensional placement problem in circuit layout, Trans. IEEE CAS-25 [1978], 208 - 214.

[10] Nakahara, H., Probabilistic signal weight in component placement, Proceedings ISCAS [1979], 677 - 680.

[11] Antreich, K.J., Johannes, F.M. and Kirsch, F.H., On the placement of components (in German), Archiv fuer Elektronik und Uebertragungstechnik 36 [1982], 1-8.

[12] Kirsch, F.H., Solution methods to the placement problem using a force model (in German), to be published in Archiv fuer Elektronik und Uebertragungstechnik 36 [1982].

APPENDIX A

The following example is provided to illustrate the structures introduced in this paper.

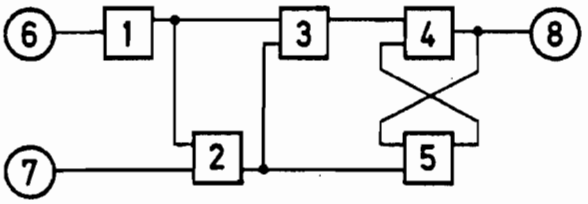


Fig. 1a Circuit topology

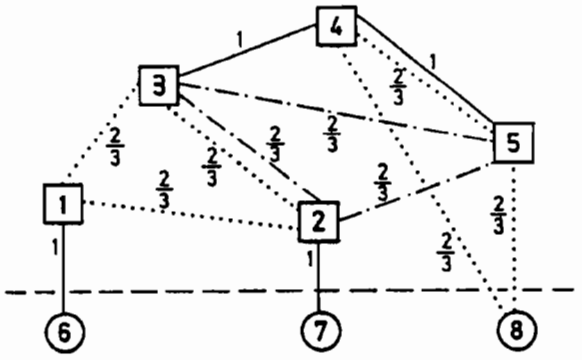


Fig. 1b Binary circuit graph

components

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix}$$

Fig. 1c Incidence matrix

Fig. 1a shows a circuit consisting of five components numbered 1 - 5 and three external connections, which are treated as components 6, 7, 8. The corresponding interconnection set N is

$$N = \{\{(1,6),(2,7),(3,4),(4,5)\},\{(1,2,3),(2,3,5),(4,5,8)\}\}$$

By replacing the 1-tuples by pairs, e.g.,

$$(4,5,8) \rightarrow (4,5),(4,8),(5,8)$$

we obtain the binary relation N_b

$$N_b = \{(1,6),(2,7),(4,8),(5,8),(1,2),(1,3),(2,3),(2,5),(3,4),(3,5),(4,5)\}$$

The weights of edges between connected component pairs are assembled in the diagonal matrix

$$\underset{\sim}{K}_{cc} = \mathrm{diag}(1, 1, \tfrac{2}{3}, \tfrac{2}{3}, \tfrac{2}{3}, \tfrac{2}{3}, \tfrac{4}{3}, \tfrac{2}{3}, 1, \tfrac{2}{3}, \tfrac{5}{3}) .$$

In Fig. 1b the set N_b of the circuit in Fig. 1a is depicted as a binary graph. To each edge the weight of the corresponding component pair is assigned. Fig. 1c shows the incidence matrix obtained from the circuit graph in Fig. 1b.

APPENDIX B

The linearized form of (21) can be determined by linearizing the SE given in Table 3, i.e., by linearizing the independent SEd in (11) and constructing the system equations using this linearization.

From (11), with the conjugate distance γ^* one obtains

$$w = k_0 \cdot \frac{\gamma}{|\gamma|} = k_0 \cdot \sqrt{\frac{\gamma}{\gamma^*}} \tag{32}$$

Letting γ_0 and w_0 represent the initial approximations of the distances and distance-oriented forces, respectively,

$$w_0 = k_0 \cdot \sqrt{\frac{\gamma_0}{\gamma_0^*}} \tag{33}$$

(32) yields

$$w = w_0 \cdot \sqrt{\frac{1+\frac{\gamma-\gamma_0}{\gamma_0}}{1+\frac{\gamma^*-\gamma_0^*}{\gamma_0^*}}} \tag{34}$$

Where the change in distance is small,

$$\left|\frac{\gamma-\gamma_0}{\gamma_0}\right| \ll 1 \tag{35}$$

the linearized form $\hat{w}$ as an approximation of w can be simply derived from

$$w \approx \hat{w} = w_0 \cdot \left(1+\frac{\gamma-\gamma_0}{2\gamma_0}-\frac{\gamma^*-\gamma_0^*}{2\gamma_0^*}\right) \tag{36}$$

Thus, $\hat{w}$ is linear in γ and γ^*.

$$\hat{w} = \frac{k_0 \cdot \gamma_0}{|\gamma_0|} \cdot \left(1+\frac{1}{2\gamma_0}\gamma-\frac{1}{2\gamma_0^*}\gamma^*\right) \tag{37}$$

This equation is independent of the metric used. Thus, if it is desireable to use a rectilinear metric, we substitute k_0 by the scale factor

$$k(\gamma_0) = \frac{k_0 \cdot |\gamma_0|}{|\mathrm{Re}(\gamma_0)|+|\mathrm{Im}(\gamma_0)|} \tag{38}$$

In constructing the solution equations, the following linear equations are substituted for the SE given in Table 3:

$$\hat{\underset{\sim}{w}}_d = \hat{\underset{\sim}{p}}_d + \hat{\underset{\sim}{P}}_{dd} \cdot \underset{\sim}{\gamma}_d - \hat{\underset{\sim}{Q}}_{dd} \cdot \underset{\sim}{\gamma}_d^* \tag{39}$$

The complex vector $\hat{\underset{\sim}{p}}_d$, the real diagonal matrix $\hat{\underset{\sim}{P}}_{dd}$ and the complex diagonal matrix $\hat{\underset{\sim}{Q}}_{dd}$ are defined as follows:

$$\hat{\underset{\sim}{p}}_d = [\ldots\ldots k_0 \cdot \frac{\gamma_{0\kappa}}{|\gamma_{0\kappa}|}\ldots\ldots]^T \tag{40}$$

$$\hat{\underset{\sim}{P}}_{dd} = \mathrm{diag}(\ldots\ldots, \frac{k_0}{2}\frac{1}{|\gamma_{0\kappa}|}, \ldots\ldots) \tag{41}$$

$$\hat{\underset{\sim}{Q}}_{dd} = \mathrm{diag}(\ldots\ldots, \frac{k_0}{2}\frac{1}{|\gamma_{0\kappa}|}\frac{\gamma_{0\kappa}}{\gamma_{0\kappa}^*}, \ldots\ldots) \tag{42}$$

Corresponding to (12), the index κ denotes all movable, disconnected component pairs.

Using (39) and the FE and DE from Table 3, the linearized form $\hat{\underset{\sim}{f}}_{m2}$ follows from $\underset{\sim}{f}_{m2}$:

$$\hat{\underset{\sim}{f}}_{m2} = \underset{\sim}{A}_{md} \cdot \hat{\underset{\sim}{p}}_d + \underset{\sim}{A}_{md} \cdot \hat{\underset{\sim}{P}}_{dd} \cdot \underset{\sim}{A}_{md}^T \cdot \underset{\sim}{z}_m - \underset{\sim}{A}_{md} \cdot \hat{\underset{\sim}{Q}}_{dd} \cdot \underset{\sim}{A}_{md}^T \cdot \underset{\sim}{z}_m^* \tag{43}$$

$$\hat{\underset{\sim}{f}}_{m2} = \underset{\sim}{p}_m + \underset{\sim}{P}_{mm} \cdot \underset{\sim}{z}_m - \underset{\sim}{Q}_{mm} \cdot \underset{\sim}{z}_m^* \tag{44}$$

Finally, we obtain the linearized system equations

$$\hat{\underset{\sim}{f}}_m = \underset{\sim}{f}_{m1} + \hat{\underset{\sim}{f}}_{m2} + \underset{\sim}{f}_{m3} = \underset{\sim}{0} \tag{45}$$

APPENDIX C

As an example the binary graph of the interconnection set N in Fig. 2, representing a placement problem with 25 movable (1-25) and 5 fixed (26-30) components, describes a circuit. The given component arrangement represents the optimum relative placement [6].

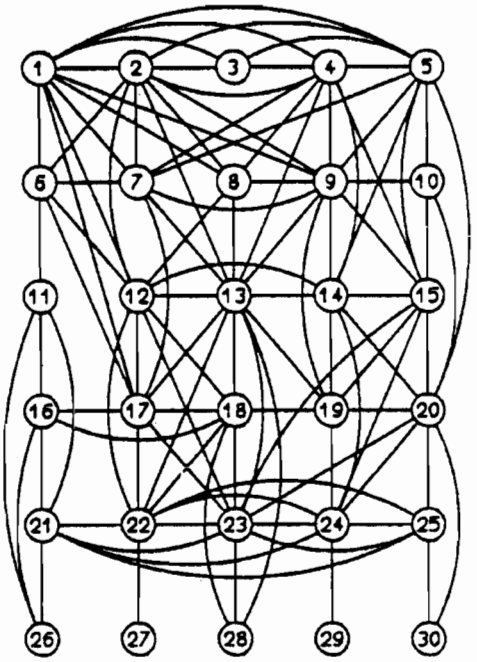


Fig. 2: Binary circuit graph with optimum component placement.

The initial placement used in the example is obtained by exchanging the components listed in Table 4.

Optimum placement of component i at location i → Location j of the initial placement			
1 → 25	8 → 12	14 → 8	19 → 9
5 → 21	9 → 7	17 → 19	21 → 5
7 → 17	12 → 18	18 → 14	25 → 1

Table 4: Initial placement.

Starting with this initial placement, the solution shown in Fig. 3 was attained in 3 iteration steps. For the sake of clarity, only the movements of components 1, 5, 21 and 25 during the first and second iteration step are shown. The final placement shown in Fig. 3 represents the optimum relative positioning of the components. In a subsequent process, the movable components, e.g., described in Chapter 4, must be assigned to definite slots. In accordance with previous experience, one can see in Fig. 3 that the components move in a direction favorable to the final placement during the first iteration step.

This example, along with others, has shown that quadratic convergence can be attained after a low number of iteration steps using the solution method applied here, despite considerable errors resulting from the linearization of repelling forces $\underline{Q}_d$ working between component pairs (see Table 5).

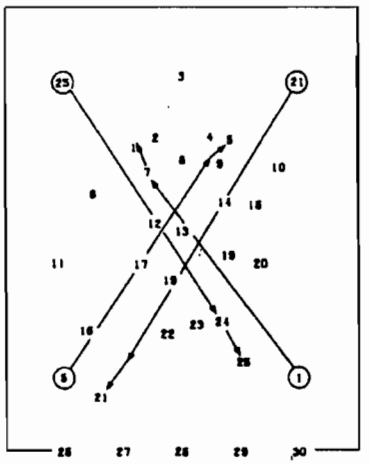


Fig. 3: Final component placement.

Iteration step μ	$\frac{\lVert \hat{\underline{w}}_d - \underline{w}_d \rVert}{\lVert \underline{w}_d \rVert}$	$\frac{\lVert \underline{z}_m^{(\mu+1)} - \underline{z}_m^{(\mu)} \rVert}{\lVert \underline{z}_m^{(\mu)} \rVert}$
1	1.015	0.9642
2	$4.029 \cdot 10^{-2}$	0.2470
3	$6.639 \cdot 10^{-4}$	$2.706 \cdot 10^{-2}$
4	$3.336 \cdot 10^{-8}$	$4.339 \cdot 10^{-4}$

Table 5: Linearization error and relative component movement.

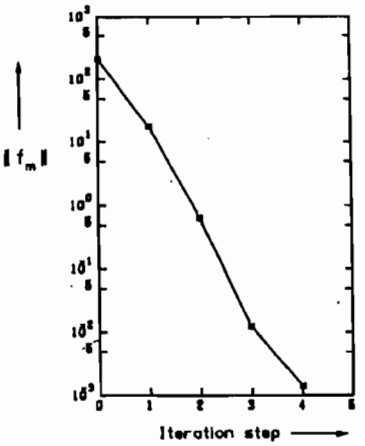


Fig. 4: Decrease of resulting forces on movable components.

The high stability of this procedure can be seen in Fig. 4, which shows the decrease in the resulting forces on the movable components after each iteration step.

TAB 45

THE PLACEMENT PROBLEM AS VIEWED FROM THE PHYSICS OF CLASSICAL MECHANICS

Neil R. Quinn, Jr.
General Dynamics Corporation
Pomona, California

Abstract

This paper presents a procedure for placing electronic components on a printed circuit board. The procedure is essentially force directed, solving a set of simultaneous nonlinear differential equations which are derived from the laws of physics. The paper derives the set of force equations, then develops an economical method for solving them.

Section I. Introduction

Just exactly what is the component placement problem with respect to printed circuit boards? We shall define it loosely as follows; given a network of components (transistors, resistors, capacitors, integrated circuits, etc.), for which we know the interconnection topology, and a copper plated fiber glass board on which to put these components; determine the optimal location of every component with respect to every other component so that the length of time spent in the interconnection (or routing) process is minimized.

This paper presents a procedure, based upon the physics of classical mechanics, which attempts to find the optimal location of all the components. The procedure minimizes a function of the interconnection topology of the system.

Section II. Component, Signal Net, Board Modeling, and Objective

A. Component Modeling

Throughout the entire procedure all components are considered to be rectangles. The lines of force to other components eminate from the geometrical center of each component.

We have three basic component types:

1. Moveable. Those components which are free to move in the X and Y directions.
2. Semi-Moveable. Those components which are free to move along only one axis.
3. Fixed. Those components which are not able to move.

B. Signal Net Modeling

All components of a signal net are assumed to be pairwise connected with a weight of 2/n where n = the number of components in the net.

The philosophy behind this weighting comes from the assumption that the more points in a net, the easier it is to connect. Hence, less significance is attached to larger nets.

C. Board Modeling

The board is modeled as a continuous rectangular plane.

D. Objective

Our goal is to derive a set of force equations, from the interconnection topology of the system, which finds the optimal relative position of each component with respect to all other components. We divided this goal into two tasks; one is to derive a set of equations for finding the optimal locations and the other is to place these equations into the proper mathematical framework for their economical solution.

Section III. Formulation of Force Equations

One's first attempt at deriving a set of force equations might be to consider all components to be connected by Hookes' Law.

For i = 1,2, . . . N

$$m_i \ddot{X}_i = F_{X_i} = \sum_{j=1}^{N} - K_{ij} * \Delta X_{ij} \qquad (1)$$

where m_i = Mass of component i
N = total number of components
$\ddot{X}_i$ = second derivative, with respect to time, of the X position of component i in a rectangular coordinate system.
K_{ij} = attractive constant between components i and j proportional to the number of signal names common to both components.
$\Delta X_{ij} = X_i - X_j$ X_i, X_j = X positions of components i and j respectfully.

and $K_{ii} = 0$

(Only equations for the X positions of the components are shown here. Similar equations exist for the Y positions.)

F_{X_i} = The net force exerted on component i by the rest of the components.

This formulation of a set of force equations has the unfortunate result that when integrated out to t = ∞, the X and Y positions of all the components are the same. In other words, everything converges to a single point.

One's next attempt at a set of equations might be to introduce some sort of repulsion between unconnected components. This seems intutively correct since one wants connected components as close together as possible and unconnected components to be forced apart. This formulation is shown in the following set of equations.

For i = 1, 2, 3, . . . N

$$F_{X_i} = \sum_{j=1}^{N} - K_{ij} * \Delta X_{ij} + \delta_{K_{ij}} * R \qquad (2)$$

where R is repulsion constant directly proportional to the maximum K_{ij} and inversly proportional to N.

$$\delta_{K_{ij}} = \begin{cases} 1 \text{ @ } K_{ij} = 0 \\ 0 \text{ @ } K_{ij} \neq 0 \end{cases}$$

This formulation keeps all the components from collapsing to a single point, but has the disadvantage of tending to spread the components out into an ellipse. This disadvantage is due to the fact that a constant repulsion force does not extend radially from each of the two components involved. Rather, it extends equally in both the X and Y directions. To overcome this, the repulsion constant is multiplied by $\Delta X_{ij}/\Delta S_{ij}$; where $\Delta S_{ij} = (\Delta X_{ij}^2 + \Delta Y_{ij}^2)^{1/2}$, which is merely the cosine of the angle the line of attraction makes with the X axis. Our modified equation now becomes:

For i = 1, 2, . . ., N

$$F_{X_i} = \sum_{j=1}^{N} - K_{ij} * \Delta X_{ij} + \delta_{K_{ij}} * R * \Delta X_{ij}/\Delta S_{ij} \qquad (3)$$

This system of equations still has two drawbacks. First, when working with circuits that have a very dense connectivity matrix (very few unconnected components), components still tend to converge to a single point. A simple example is shown in Figure 1.

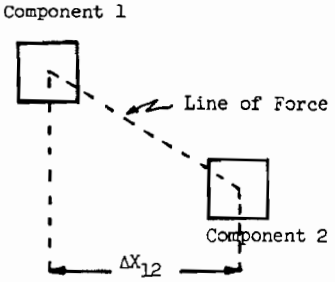


Figure 1.

If (3) is used to solve this problem, the answer will result in $\Delta X_{12} = 0$, or the two components sharing the same physical location. To overcome this dilemma, ΔX_{12} must be equal to the distance between the inside perimeters of the components as shown in Figure 2.

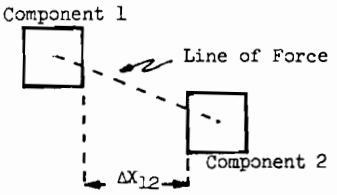


Figure 2.

Using this new value for ΔX_{12} we get the solution shown in Figure 3.

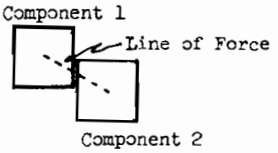


Figure 3.

The second drawback of (3) becomes evident when the circuit board has one or more sets of connector pins (Figure 4). If we use (3)(with no repulsion between connectors and unconnected components) to solve the system shown in Figure 4, the result might look something like that shown in Figure 5.

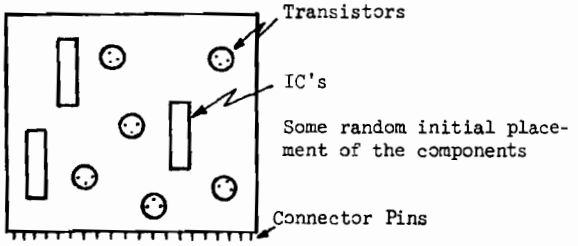


Figure 4.

Here we see that the solution places the components around the connector, some lying off the actual circuit board.

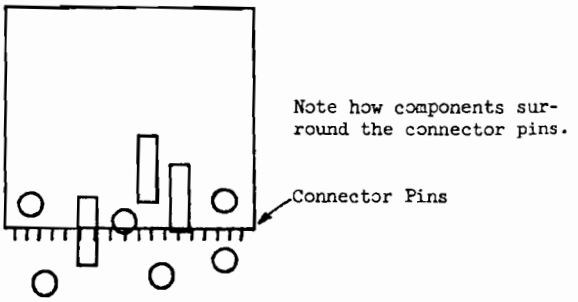
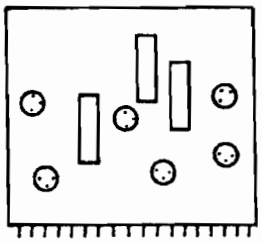


Figure 5.

Connectors are handled just like any other element, except that their mass is considered to be infinite. Therefore, they never move.

For this reason, the moveable components surround the connector, with those components that are highly connected to the I/O pins lying almost on top of them and those components that are unconnected to the I/O pins lying above and below them.

If we blindly move all the components up to the center of the board (Figure 6), we do not get a good placement, since those components that must be connected to the I/O pins fall in the center of the group instead of adjacent to the connector pins.



Components of Figure 5 just moved up to lie in the center of the board.

Figure 6.

As will be shown later, it is necessary to keep the center of mass of the movable components affixed to some predetermined physical location (usually the geometrical center of the board).

If we include repulsion between the connector and unconnected components, then none of the components will overlap the connector. However, this type of repulsion does not insure that the center of mass of the components will remain fixed.

To overcome this problem we again turn to the laws of physics, using those laws which pertain to the center of mass of a system. We can consider all the movable components to be a closed system, acted upon by external forces exerted by the fixed components. By subtracting out the net effect of the external forces, thus leaving the center of mass of the movable components fixed, we arrive at our final set of force equations.

For i = 1, 2, . . ., N

$$F_{X_i} = \sum_{j=1}^{N} \left[-K_{ij} * \Delta X_{ij} + \delta K_{ij} * R * \Delta X_{ij}/\Delta S_{ij}\right] - F_{CMX} \qquad (4)$$

Where $F_{CMX} = \left[\sum_{i=1}^{M} \sum_{j=M+1}^{N} -K_{ij} * \Delta X_{ij} + \delta K_{ij} * R * \Delta X_{ij}/\Delta S_{ij}\right] /M$

M = the number of movable components
N-M = the number of fixed components
and ΔX_{ij} is the distance between the inside perimeters of the components.

Section IV. Solution of (4)

Now that we have arrived at a suitable set of force equations describing the placement problem, we must investigate ways of solving these equations economically.

Solution Method 1.

The first, and most obvious way, is to consider these equations to be a set of simultaneous nonlinear differential equations describing the second derivatives, with respect to time, of the component positions.

Differential Equations Formulation:

$$m_i \ddot{X}_i = \sum_{j=1}^{N} \left[-C_{ij} * \Delta \dot{X}_{ij} - K_{ij} * \Delta X_{ij} + \delta K_{ij} * R * \Delta X_{ij}/\Delta S_{ij}\right] - F_{CMX} \qquad (5)$$

where $\ddot{X}_i$ = the acceleration of component i in the X direction.

$\Delta \dot{X}_{ij} = \dot{X}_i - \dot{X}_j$

$\dot{X}_i$ = the velocity of component i in the X direction.

and C_{ij} = a damping constant that ensures that the system comes to a non-oscillatory steady state solution.

$= 2*\sqrt{K_{ij}}$ for critical damping.

These equations may be integrated by any standard integration formula. We have used a first order Euler formula with much success.

This solution method does have one drawback, and that is execution time. Since the solutions are exponentials, we encounter the law of diminishing returns in looking for the steady state solution (T = ∞).

Solution Method 2.

Since we know that $\ddot{X}_i$ and $\dot{X}_i = 0$ at t = ∞, we can formulate the equations as follows and solve for the values of X_i algebraically (Newton's Method).

Algebraic Formulation

$$0 = \sum_{j=1}^{N} \left[-K_{ij} * \Delta X_{ij} + \delta K_{ij} * R * \Delta X_{ij}/\Delta S_{ij}\right] - F_{CMX} \qquad (6)$$

This method tends to work quite well for small systems (less than 50 components), but becomes computationally infeasible for larger systems because of storage requirements and round-off errors.

Solution Method 3.

In an attempt to overcome the difficulties of methods 1 and 2, we discovered the following approach.

The solution to our problem actually corresponds to the state of zero potential energy of the system.

Recalling that the potential energy is defined as

$$V = -\sum_{i=1}^{N} \int_{X_{i_{t=0}}}^{X_{i_{t=\infty}}} F_{X_i}\, dX$$

we see that F_{X_i} is really the partial derivative of V with respect to X_i. Therefore, we can formulate this problem as one of finding the unconstrained minimization of the potential energy function V, given the gradients of V, which are the F_{X_i}'s.

We have set $V = \sum_{i=1}^{N} F_{X_i}^2$ and used the Fletcher - Reeves [Reference 3] unconstrained minimization procedure. We chose the Fletcher Reeves method because of its low storage requirements.

This approach gives excellent results in approximately 1/10 the time required by the differential equation method.

Section V. Post Processor

After solving (4) for a steady state solution we see that the majority of the components are overlapping. This apparent conflict is overcome quite easily in one of two ways.

Method 1.

For boards containing primarily IC's, which must go in predetermined slots, we use Munkres' [Reference 4] algorithm to solve the problem of assigning the IC's to slots. As a cost function, we use the Euclidian distance from each IC to each slot. This preserves the relative position of each of the components, since (4) keeps the center of mass of the components fixed at the geometrical center of the slots.

Method 2.

For boards containing discrete components, (transistors, resistors, etc.) we alleviate the overlap by moving those components farthest away from center of mass radially outward by an amount equal to ΔS_{ij}. This again preserves component relative position as stated above.

Section VI. Experimental Results

The evaluation of any placement algorithm is a difficult process, to say the least. We have chosen the following two methods for evaluating our algorithm.

1. Determination of whether or not the algorithm finds a known optimum placement.
2. Manual evaluation, by means of a graphics facility which enables one to look at how the various components are interconnected.

Several lattice structures, of the type shown in Figure 7, were derived for use with method 1. The signal net lists used to connect the components are shown in table 1. Each structure was tried several times with different random initial placements for the elements, and each time the algorithm found the optimal solution. Structure 1 was also tried using random weights, on the interval (1-100), between connected elements. Again, the optimal solution was found.

This type of evaluation is most encouraging.

A large number of production type P.C.B.'s, varing in size from 30 to 100 elements, and containing a mixture of I.C.'s and discrete elements, have been evaluated by competent packaging engineers using method 2. They concluded that they would not change the relative positions of any of the components on any of the boards. They also commented that the solutions found are what they would strive for if they were placing these boards manually.

Section VII. Conclusions and Comments

It is our conclusion that this type of parallel force directed placement procedure exhibits very favorable results is finding correct solutions to known optimal placements and in finding the same type of solution that a man seeks when performing placement manually.

The algorithm bears some similarity to works by References [1], and [2]; however, the major differences are:

(1) a sound mathematical foundation for all the force equations and solutions used.
(2) the force constraining the center of mass of all moveable components to remain about the center of the board.
(3) radial repulsion between unconnected components.

Bibliography

1. C.J. Fisk, D.L. Caskey, and L.L. West, "Accel: Automated Circuit Card Etching Layout." Proc. IEEE, Vol. 55, No. 11 (1967), pgs. 1971-1982.

2. F.T. Scanlon, "Automated Placement of Multiterminal Components", Proc. Design Automation Workshop, July 1971.

3. R.Fletcher, and C.M. Reeves, "Function Minimization by Conjugate Gradients" Computer Journal, Vol. 7, 1964, pgs. 149-154.

4. J. Munkres, "Algorithms for the Assignment and Transportation Problems", J. Soc. Industrial Applied Math. Vol. V (March 1957), pgs. 32-38.

Acknowledgements

The author is grateful to Dr. Melvin A. Breuer for his assistance and suggestions.

SYMETRICAL RECTANGULAR LATTICE OF COMPONENTS

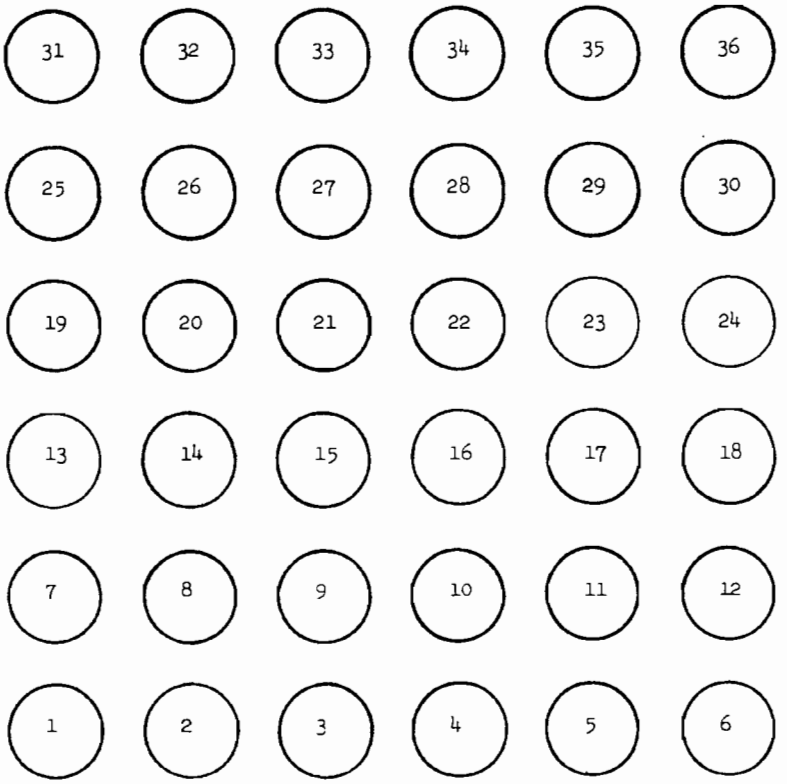


Figure 7.

Table 1. Signal Net, Component Lists Used.

Structure 1.	Signal Net	Components Contained In Signal Net
	1	1, 2
	2	2, 3
	3	3, 4
	.	
	.	
	.	
	6	1, 7
	7	2, 8
	.	
	.	
	.	
	12	7, 8
	13	8, 9
	.	
	.	
	.	
	59	34, 35
	60	35, 36

Table 1. (Continued)

Structure 2.	Signal Net	Components Contained In Signal Net
	1	1, 2, 7, 8
	2	2, 3, 8, 9
	3	3, 4, 9, 10
	.	
	.	
	.	
	6	7, 8, 13, 14
	.	
	.	
	.	
	.	
	25	20, 30, 35, 36
Structure 3.	1	1, 2, 3, 7, 8, 9, 13, 14, 15
	2	2, 3, 4, 8, 9, 10, 14, 15, 16
	3	3, 4, 5, 9, 10, 11, 15, 16, 17
	4	4, 5, 6, 10, 11, 12, 16, 17, 18
	5	7, 8, 9, 13, 14, 15, 19, 20, 21
	.	
	.	
	.	
	.	
	16	22, 23, 24, 28, 29, 30, 34, 35, 36

TAB 46

LAYOUT DESIGN AND VERIFICATION
T. Ohtsuki (Editor)

Chapter 2

PARTITIONING, ASSIGNMENT AND PLACEMENT

Satoshi GOTO, Tsuneo MATSUDA

C&C Systems Research Labs., NEC Corporation
Kawasaki 213, Japan

1. INTRODUCTION

Layout design for an electronic equipment requires the partitioning and assignment of logic circuits to physical design modules, placement of these design units onto larger function modules and finding the routing patterns for interconnection. According to the physical hierarchy of hardware elements, modules correspond to function blocks, LSIs, printed circuit boards and backboards.

The current integrated circuit approach usually results in three-level backplanes. On the lowest level, sets of logic elements form modules. Interconnected sets of modules define boards, and interconnected sets of boards define the backplane.

In this VLSI era, modules correspond to VLSIs, ICs or discrete components. VLSIs have tremendous number of gates in themselves, and usually have hierarchical structures. On the lowest level, sets of logic elements form blocks. Interconnected sets of cells define macros, and interconnected sets of macros define a VLSI chip. See Fig. 1 and 2.

Although the problems involved in partioning, assignment, placement and routing are closely related, they have been treated separately because of the inherent computational complexity of the total problem. This chapter deals with partitioning, assignment and placement problems, since other chapters in this text deal with examing the routing problem.

The partition problem is viewed in this chapter as a design process which a module is constructed from submodules at a given hierarchy level. The assignment problem is to find one-to-one correspondence between logic circuit elements and placeable modules. The placement problem is to find appropriate locations for individual modules on the chip or boards.

These problems are considered to be hard combinatorial problems, or NP-complete problems from the computational complexity point of view, in the sense that the computation time required to obtain the real optimum solution increases in exponential order when the problem size increases. And also, actual problems have various kinds of object functions with a lot of restrictions.

Unfortunately, no methods exist which guarantee an optimum solution for the real-world large scale problems. Hence, algorithms based on heuristic rationales have been employed. All of these algorithms are either constructive or iterative. The constructive algorithms produce a solution using heuristic rules, often in sequential, deterministic

manner. On the other hand, the iterative algorithms improve a solution by repeated modification of it. The automated layout design system usually has the above two algorithms, i.e., an initial solution is obtained through a constructive algorithm and the solution is improved gradually by an iterative algorithm.

Although many intensive efforts have been carried out in these years, the results obtained from the automated layout design systems are still quite discouraging, when compared to manually designed ones. This is because a human being or circuit designer can well understand the given electronic circuits, and he can find a good solution by exploiting his knowledge and intuition. Several approaches have been introduced to cope with this situation. The man-machine interaction system, high level function description or artificial intelligent approaches are notable approaches in this field.

This chapter is intended to give the current status of the partitioning, assignment and placement solving methods with tutorials. The emphasis is on rather new methods which seem to have a potential for solving some of the current problems in a practical manner.

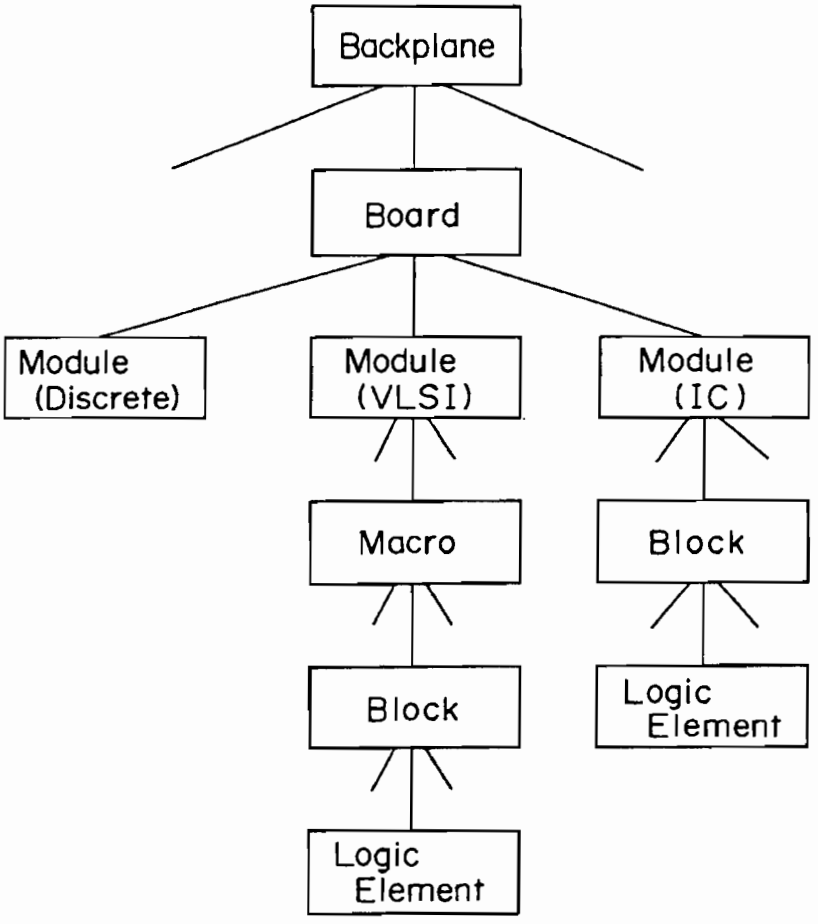


Fig. 1 Physical Hierarchy

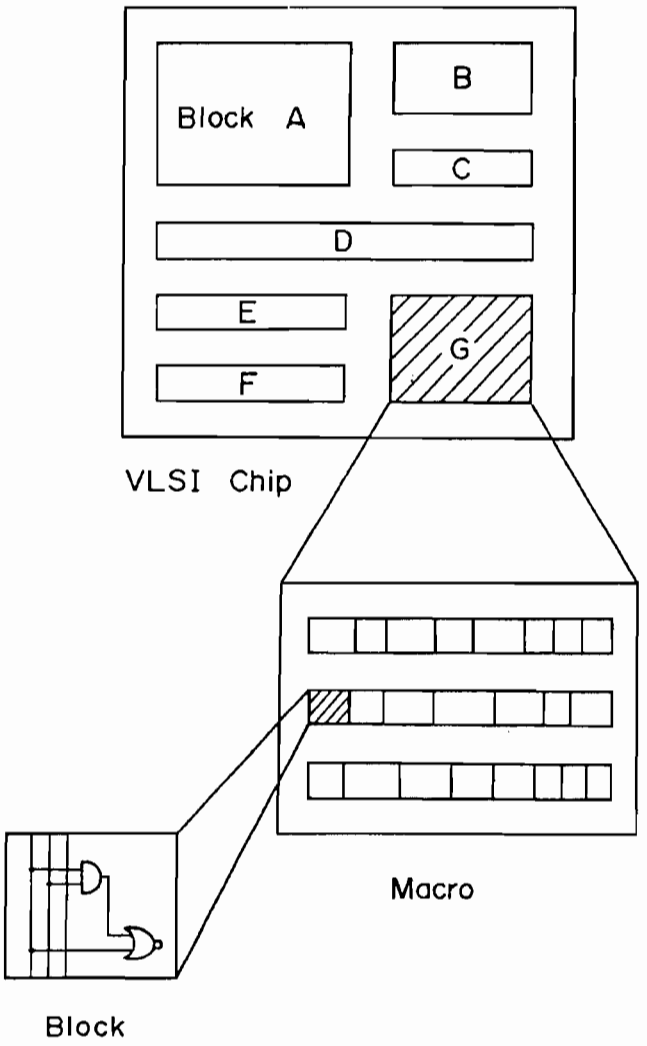


Fig. 2 VLSI Chip Hierarchy.

2. PARTITIONING PROBLEM

2.1 Introduction

Partitioning is a process of creating the physical hierarchy of the hardware elements used to realize the design. It is essentially the subdivision of logic complexes into smaller subcomplexes. The physical structure has, in general, a hierarchical structure, with components such as LSIs, printed circuit boards, backboards and systems. The partitioning can define which LSIs contain what logic, and which boards contain what LSIs under several constraints. See Fig. 1 for the hierarchical structure.

There are two different applications for partitioning logic circuits. One is for circuit packaging, which divide a complex of logic circuit into subcomplexes, which can be fitted into the packaging structure used for the complexes. The other one involves a circuit layout for a VLSI chip. This partitioning is used in the layout of a complex, and is preparatory for the placement. Logic partitioning will find the global structure for a logic complex more easily than direct placement algorithms.

Parameters or constraints are associated with a partitioning problem which measure the solution of a partitioning problem.
They are as follows.

(a) The size or area of each partition element is restricted.
(b) The number of external connections for each partition element is restricted.
(c) The maximum delay time through the circuits must not exceed a given time. Any external connection contributes a longer signal transmission delay time than an internal connection.
(d) A certain signal net must be externally available for testing purposes.

The above constraints are rather complicated and sometimes not well defined. In many practical cases, expert designers intuitively give good solutions based on their experience, taking the complicated situation into account. At this moment, very few automatic programs are practically used for partitioning because of the extreme complexities of the problems involved.

In the following sections, we look upon the partitioning problem from a mathematical point of view and give the formulation and its solving methods. The latest overview on this problem is given by Kodres [1], and we give considerably new results since then.

2.2 Mathematical Formuration

Any patitioning problem must consider two parameters, size and external connection requirements. See (a) and (b) in the previous Section. Neglecting, for the moment, any other parameters (c) and (d), we may define a partition problem as follows.

Let V be a set of nodes, and assign to each node $v \in V$ size S(v), and the number of nets connected to v, N(v), respectively. With each subset Vi of node set V, we define the number of nets, I(v) for $v \in V_i$ such that nodes adjacent to v belong only to Vi. The number

$$E(v) = N(v) - I(v) \tag{1}$$

denotes the number of external nets for a subset Vi. The size and external connection limits are denoted by S, and E, respectively. The partitioning problem is to find a family of subsets of V such that

$$\sum_{v \in Vi} S(v) \leqq S \tag{2}$$

$$\sum_{v \in Vi} E(v) \leqq E \tag{3}$$

where for $i = 1, 2, \cdots, n$,

$$\bigcup_{i=1}^{n} Vi = V \tag{4}$$

$$Vi \bigcap Vj = \phi \tag{5}$$

If the objective is to minimize the total number of external connections between partition sets, Eq. (3) is replaced by

$$T = \frac{1}{2} \sum_{i=1}^{n} \sum_{v \in Vi} E(v) \tag{6}$$

$$T \rightarrow Min$$

It should be noted that, if a net common to more than two nodes exstis, the calculation of E(v) has to be modified. N(v) and I(v) have different values according to the way net connection patterns are set up.

Consider an example circuit shown in Fig. 3. with node set $V=\{1, 2, \cdots, 12\}$. We assume that $S(v)=1$ for $v \in V$, and want to partition the circuit into 3 sub-circuits with constraint $S=4$.

First, we want to have a partitioning for the limit of external connection $E=4$. A feasible partition is obtained by $V1=\{1, 4, 6, 8\}$, $V2=\{2, 3, 5, 12\}$ and $V3=\{7, 9, 10, 11\}$, shown in Fig. 4. However, this partition does not satisfy the minimum number of total external connections. On the other hand, the partitioning, $V1=\{1, 2, 3, 12\}$, $V2=\{5, 7, 10, 11\}$ and $V3=\{4, 6, 8, 9\}$, shown in Fig. 5, does satisfy it, and also satisfies the limit of external connection $E=4$. In many cases, the partition with the minimum number of total external connections between partition sets satisfies the limit of external connections for each partition set.

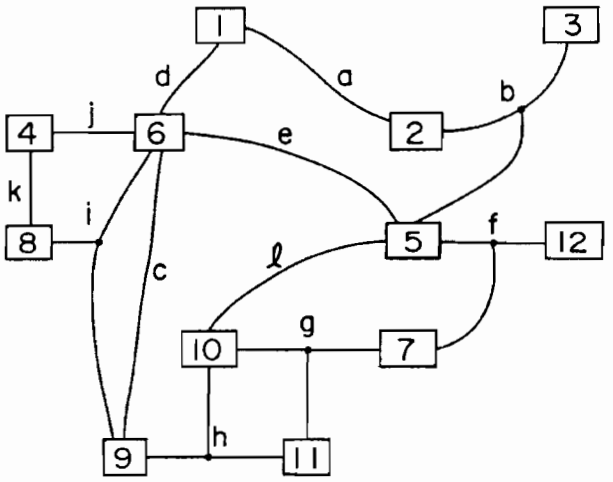


Fig. 3 An Example Network.

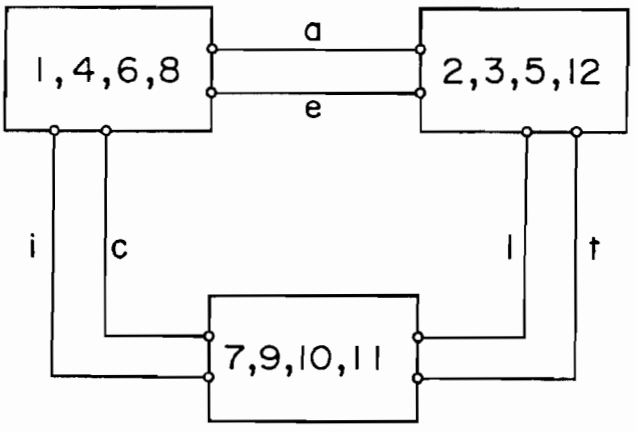


Fig. 4 Partitioning Result with External Connection $E=4$ Limit.

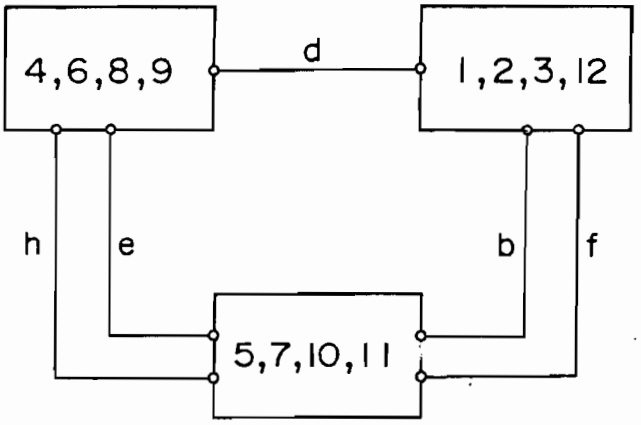


Fig. 5 Partitioning with the Minimum Number of Total External Connections, T=5.

2.3 Partitioning Algorithms [2-7]

There exist two kinds of heuristic algorithms, constructive and iterative improvement ones. The constructive algorithm produce a solution in a sequential and deterministic manner. On the other hand, the iterative one improves a solution by iteratively reducing the solution.

2.3.1 Constructive Algorithms

This method selects nodes, one at a time, based on an evaluation function, called IOC, which measures net connectivity to nets already or not yet selected, and then decides which partitioning set the selected node should belong to. The evaluation function is, in general, to be the number of nets connected to already selected nodes minus the number of nets connected to not yet selected nodes. The node with the highest value is the logical candidate to be selected. The selected node belongs to a partition set, which results in the minimum number of total external connections at this stage under the limit of external connection for each partition set.

This method starts with a seed or set of seeds for each partition set. A start with only one seed is called, "Max Conjunction-Min Disjunction Method," and a start with a set of seeds is called, "Clustering Method." However, these two methods do not involve appreciable differences between each other.

As an example, start with a seed, node 3 in Fig. 3, and obtain a partition in Fig.6 by applying the above algorithm to minimize the total number of external connections. Here, the number is 7 and, of course, it is not the optimum solution.

2.3.2 Iterative Improvement Algorithm

The method starts with an initially given partitioning result and improves it by a local transformation. The resultant partition is locally optimum for each initial partition in the sense that there does not exist a partitioning with a smaller cost for the given transformation, or modification rule and stopping rule.

We can generate a large number of locally optimum solution by generating initial partitions randomly, and thereby increase the probability that, at best, one of these solutions is close to the global optimum. In general, a more complicated transformation reaches one locally optimum with a better value in a greater computation time than a simpler one. In order to find a better solution within a given amount of computation time, the key problem is to find a suitable transformation.

For this type of problem, several discussions have been done as local optimization techniques which can apply to many other discrete optimization problems. We will discuss it in details in the placement section of Sec. 4. The result can be directly applied to a partitioning problem. Here, we describe a so-called, "Group Migration Method," proposed by Kernighan and Lin [5], which was developed in 1970 and is still one of the best methods in the past 15 years.

Interchange methods try to exchange nodes between partition sets. Pairwise interchange is applied to two nodes and triple interchange is to three nodes. An interchange operation is accepted if the resultant solution is a better one, and is not acepted if it is not. The new configuration is substituted for the old one.

Group Migration Method is used in an attempt to interchange more than two nodes simultaneously and improves a solution efficiently. The method is particularly well suited for bisection problem, where the circuit is divided into two subsets. Consider two initially constructed sets A and B, $A \cup B = V$ to start the process. If node $a \in A$ is exchanged with node $b \in B$, calculate the number of external connections gained by the interchange of a and b. This gain is denoted by g. If $g > 0$, then the interchange is beneficial, i.e., the total number of connections is reduced.

Let $a_1 \in A$ and $b_1 \in B$ be the nodes which result in the biggest gain , g., among all pair interchanges. Remove a_1 from A and b_1 from B, and repeat the process to find $a_2 \in A - a_1$ and $b_2 \in A - b_1$. By repeating this process until A and B are reduced to a null set, we get sequence $a_1, a_2, a_3, \cdots$; $b_1, b_2, b_3, \cdots$, and $g_1, g_2, g_3 \cdots$. We find a value k such that

$$G = \sum_{i=1}^{k} g_i \tag{7}$$

is maximum, then move $a_1, \cdots a_k$ to B and $b_1, \cdots, b_k$ to A. If G is positive and k is greater than zero, repeat the above process. Otherwise, stop the procedure.

For example, starting with a solution of Fig. 6, we obtain a solution of Fig. 4 by moving nodes 2, 7 and 8 simultaneously. However, in order to get a solution of Fig.5, we have to move nodes 4 and 6, 10 and 11, and 3 and 12 at the same time, which requires very complicated transformations.

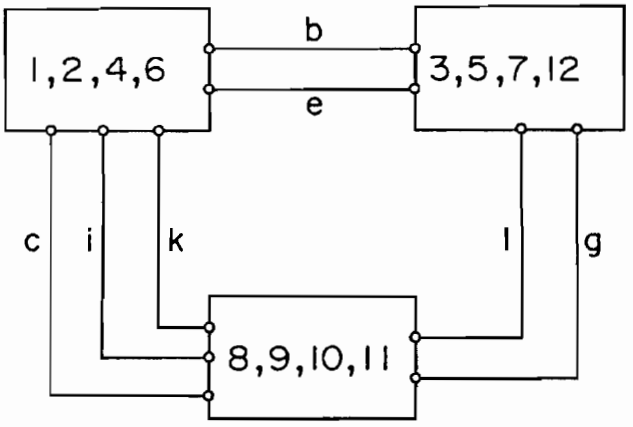


Fig. 6 A Partitioning Result using Constructive Algorithm, T=7.

2.3.3 Other partitioning methods

Several new approaches have been proposed up to now. Some approaches displayed remarkable results and some are now under investigation. Unfortunately, we cannot say clearly which method is better than the others because of the shortage of sufficient experimental results. We just show interesting and recently developed ones for the reader's reference.

(1) Metric Allocation Method

This method attempts to find a metric allocation other than the graph structure on the nodes of the graph, which reflects the connectedness of the graph. As a metric, it uses an electrical analog for the network [8] or it calculates eigenvalue or eigenvector to obtain partitioning solutions [9].

(2) Function Oriented Method

This method utilizes the information of logic circuits to determine a good partitioning. In [10], hierarchical design information is used to locate sets of identical logic, which can be realized as repeated physical entities to increase functional partitioning, and to divide the partitioning problem into multiple smaller problems. In [11], an algorithm is proposed for partitioning a behavioral hardware description written in the ISPS computer hardware description language. The partitioning is carried out before the actual registers, processing elements, and interconnections have been chosen, so that the partitioning information can be used to guide the design of the data path structure.

3. Assignment Problem

An assignment is the phase of building a circuit of modules with size, shape and internal structure for a given logic circuit. The logic circuit is usually a schematic diagram where logic blocks, such as NAND, NOR gates or Inverters are connected by wires. The module circuit has to satisfy given circuit performance requirements such as delay time, power consumption or number of input-output terminals.

There exist different type assignment problems according to the different technologies. We will describe two typical problems [12], which plays important roles in the circuit layout.

3.1 Printed Circuit Board Problem

PCB design problem requires assigning abstract blocks to IC packages of TTL (Transistor-Transistor-Logic) circuits. Various kinds of IC packages are provided by IC makers, and can be used by selecting suitable ones. For example, IC package ××1 contains four two-input NAND gates and ××2 contains contains three two-input NOR buffers. If the design circuit includes 17 NAND gates and 20 NOR buffers, the board requires at least five ××1 and seven ××2 IC packages.

Some gates are left unused for later repairs or for circuit modifications. The assignment or selection rules are applied by taking into account easy routability and minimum signal transmission delay time.

The usual assignment algorithm is based on the following rules [12].

① Assign all ICs which contain only one logic block or gate.
② If an IC package has an available gate, use it for the gate which has the most connection in common with gates already assigned to this IC package.
③ If there is no partially used IC package, take the next unused package, and assign the first gate randomly.

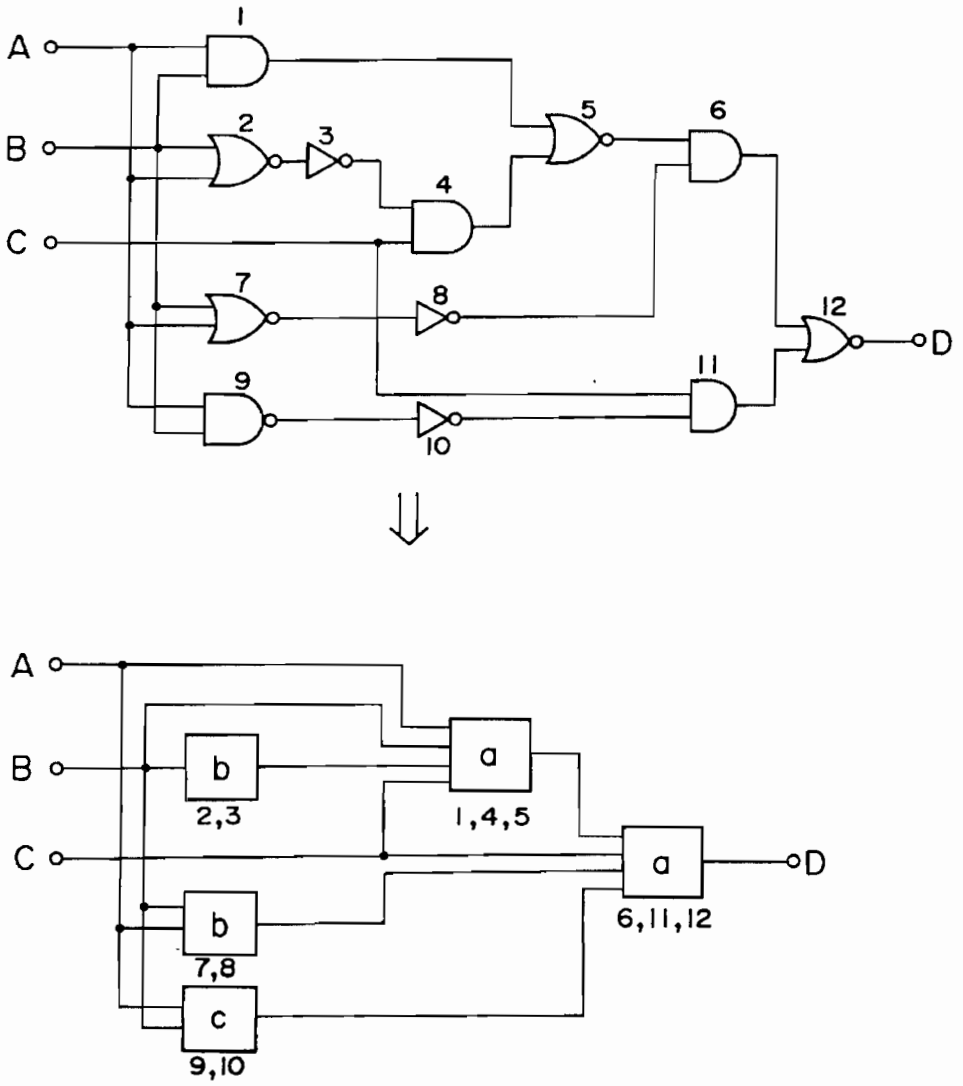


Fig. 7 Assignment from Logic Blocks to Circuit Modules.

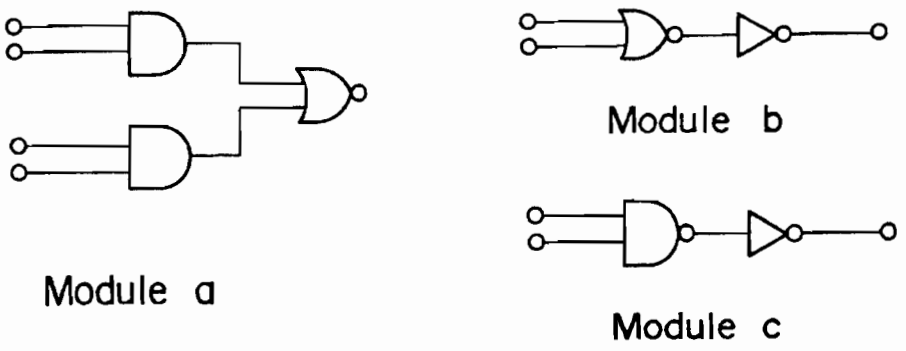


Fig. 8 Module Libraries Example.

3.2 LSI Layout Problem

In the gate-array or standard cell LSI design, fundamental logic elements, called circuit modules, such as NAND, NOR or FLIP-FLOP, are designed by giving interconnections between transistors or resistors in advance to meet the customer's LSI design requirements and stored in the library. The number of circuit modules in the library is usually between 100 and 200. Most logic blocks correspond to circuit modules in the library on a one-to-one basis, except that several circuit modules are available for each logic block. Such circuit modules have identical logic, but their output power and sizes are different. Also, certain circuit modules may be available in several shapes or terminal arrangements. Figure 7 shows an assignment example for use from logic blocks to circuit modules. The logic circuit with 12 blocks is converted to the module circuit with 5 modules by using 3 library modules, as shown in Fig. 8.

The assignment algorithm requires the following two steps.

① Find which part of logic circuits corresponds to library modules to have the same function.
② Determine which library module should be used to meet the required power, delay time and shape requirements.

The usual assignment algorithm is carried out in a straight forward way, picking up logic blocks one by one, according to the connectivities, and finding a suitable circuit module to meet function and performance requirements.

The above algorithms are based on constructive methods and are very simple to implement. There might be more complex ones, such as iterative improvement methods, which could be used to achieve better results, however no practically efficient algorithms have been found up to now, as far as the author knows. After the automatic assignment is accomplished by computer program, the results are usually improved on graphic stations by expert circuit designers.

3.3 Other Assignment Problems

In an automatic routing for a gate array LSI, a problem arises in regard to which one of the logically equivalent terminals is to be used to allow a signal net to be connected to any one of such terminals. There is also a problem regarding which vertical track in a feed-through cell is to be used for a signal net assigned to the feed-through cell. An efficient heuristic algorithm is proposed in [13], which tries to achieve 100% realization of the wiring requirements.

4. PLACEMENT PROBLEM

4.1 Introduction

For each level of the hierarchical structure shown in Fig. 1, the placement problem has to be solved. Board placement has to be solved for backboard design. Module placement for board has to be determined, macro placement for VLSI must be settled, and block placement for macro or LSI must be accomplished.

The board placement, the module placement and the block placement problems can be treated in similar ways, since each component, in general, has the same size or the same height, even though components have different widths. Figure 9 shows a module placement example. Figure 10 and 11 are block placement examples, respectively. On the other hand, in the macro placement problem, individual components have various kinds of shapes or sizes, from larger to smaller, since it corresponds to RAM, ROM, PLA or a set of random logics, as shown in Fig. 2.

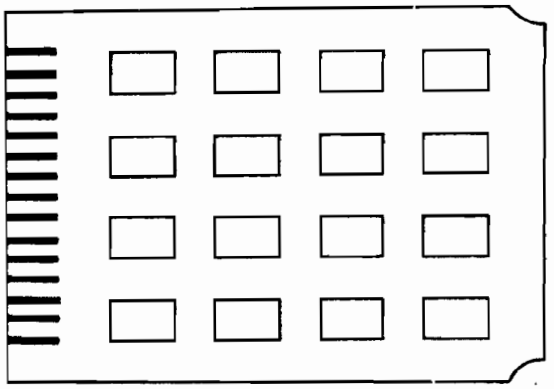


Fig. 9 Module Placement.

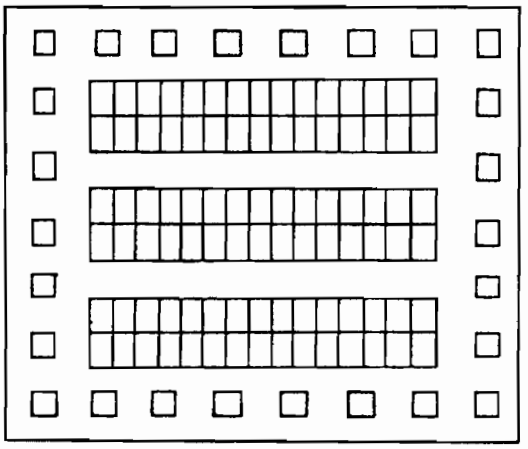


Fig. 10 Block Placement (Gate Array LSI).

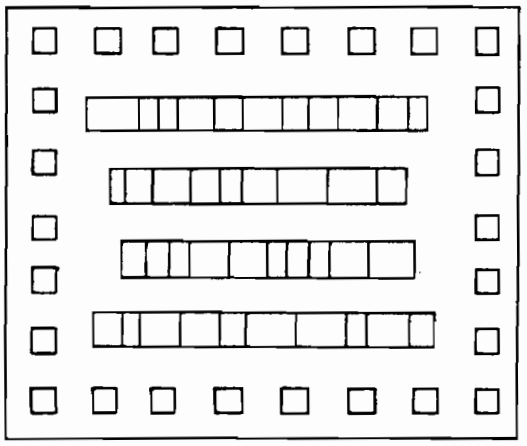


Fig. 11 Block Placement (Poly Cell LSI).

The macro placement problem has to be treated in a different manner from the others because of the irregular component shape. The module placement problem has been studied for many years to make the package design automatically, and its result can be applied directly to the block placement for macro or LSI. Particularly, the block placement problem in gate-array LSIs or poly-cell LSIs has been of great interest to LSI CAD engineers recently. The problem is of particular significance in the present day design of LSI chips, as a huge number of components has been mounted on one chip, i.e., 1000~10000 components. Therefore, efficient new algorithms have to be devised to meet such a situation.

The macro placement is a new problem which did not appear in the IC eras. The number of components, or gates per chip, has become too big to handle simultaneously, the hirarchical design methodology has been introduced.

For our convenience, we use, in the following section, the terminologies, blocks and a chip as the placement problem. Blocks are assumed to have various heights or sizes.

The layout problem is usually divided into two stages and solved as different problems, namely placement and routing. In the placement stage, locations of individual blocks, are decided on a chip to facilitate routing. In the routing stage, interconnections to external leads, called pins, are made in such a manner as to satisfy various physical constraints.

The true objective of the placement phase is to achieve 100% routing within a given area.
Such an objective is not mathematically well defined.

We are not able to know whether one placement result is good or not, until trying the actual routing. We replace the true goal by a simplified objective which is easy to calculate by computer. When we choose one objective, we hope that a solution to optimize the objective will lead to a high routability result. Until now, the following three objectives have been proposed to minimize:

(1) Total routing length.
(2) Maximum cut line.
(3) Maximum density.

For many years, the total routing length objective has been employed. However, this objective sometimes generates too crowded an unroutable routing area, and the second or the third objective has been introduced. In the following sections, the relations between these objectives are discussed and efficient algorithms for each objective are described.

Prior to the placement stage, block design and the assignment of logic gates into blocks have been accomplished. Logic elements, called blocks, such as AND, OR, or FLIP-FLOP, are designed by giving interconnections by conductors between transistors or resistors. The whole logic function of a given circuit can be realized by connecting between block terminals and external pads on the routing area. Therefore, circuit data is reduced to between-the-block connections, which are called a set of signal nets.

The latest overview on this problem is given by Hanan and Kurtzberg in [14]. They also presented valuable papers with many experimental results. In the following, considerably new results since [14] are described.

4.2 Interconnection Rules

In the placement stage, locations of individual blocks are decided on a chip to facilitate routing. Without trying the actual routing or geometric routing, it is necessary to decide whether one placement result is good or not at the placement stage. We need a simpler routing method at the placement stage, which can be incorporated into the

routing program and reflect the actual routing with good approximaion. Sometimes, the technology used in the production dominates the interconnection rules, which must be satisfied in the actual routing. The followings are examples of interconnection rules.

(a) Minimum Steiner Tree
Minimum length tree, whose vertices include block terminals and junction points.

(b) Minimum Spanning Tree
Minimum length tree, whose vertices include only block terminals.

(c) Minimum Chain
Minimum length tree, which has at most two edges incident to any vertex.

(d) Minimum Source-to-Sink Connection
Minimum length tree, which has a connection from the source to each sink.

Figure 12 shows the examples. A vertex with ⊗ or × symbols represents the source terminal and sink terminal, respectively. A vertex with ○ symbol represents the junction point. The lengths of each tree are $l=10$, 11, 12, and 15. In general, the shortest tree is the Minimum Steiner Tree, while the longest one is the Minimum Chain or the Minimum Source-to-Sink Connection. Except for the Minimum Steiner Tree, the optimum solution is obtained easily, i.e., in a polynomial order computation time. The problem of finding the Minimum Steiner Tree is proven to be on NP-complete problem, which possibly requires exponential computation time, when the problem size becomes bigger. A simple heuristic procedure is practically introduced to obtain the Minimum Steiner Tree.

Because the above four trees require some computation effort, even with simple heuristics, simpler approximation methods are introduced, called "Complete Graph Method or Half Perimeter Method." As far as the routing length is concerned, these simpler methods are considered sufficiently capable of reflecting achieving the goal.

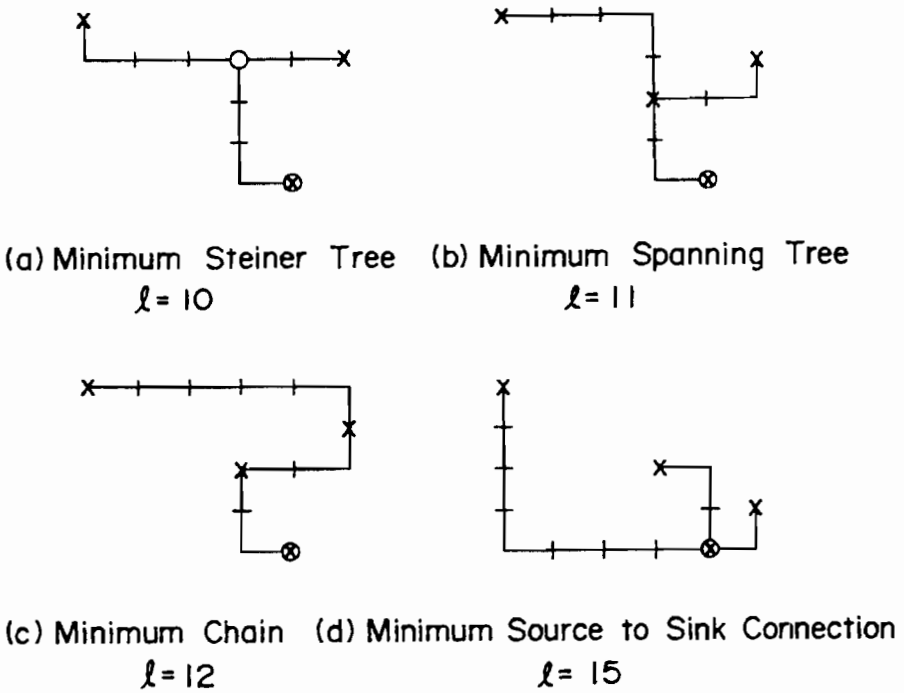


Fig. 12 Interconnection Rules.

(e) Complete Graph Method [15].
Let a signal net be common to ρ blocks, the length of a signal net is calculated as

$$\frac{2}{\rho}\sum_{i=1}^{1/2\rho(\rho-1)} Length\ of\ between-the-block. \tag{8}$$

The intuitive rationale for this is that $2/\rho$ is the fraction of the $1/2\rho(\rho\text{-}1)$ edges needed to connect a signal net of size ρ. The length for the same example, in Fig. 12, is calculated as $2/4(8+3+4+5+3+6)=14.5$

(f) Half Perimeter Method
The length is defined as half-perimeter of the smallest rectangle which encloses the blocks in the signal net.

The length is calculated as $l=9$ for the same example, since the horizontal and vertical lengths are 4 and 5, respectively. These values are shrter than those of the Minimum Steiner Tree. However, in general, for a signal net with up to 3 blocks, the length is equal to the length of the Minimum Steiner Tree.

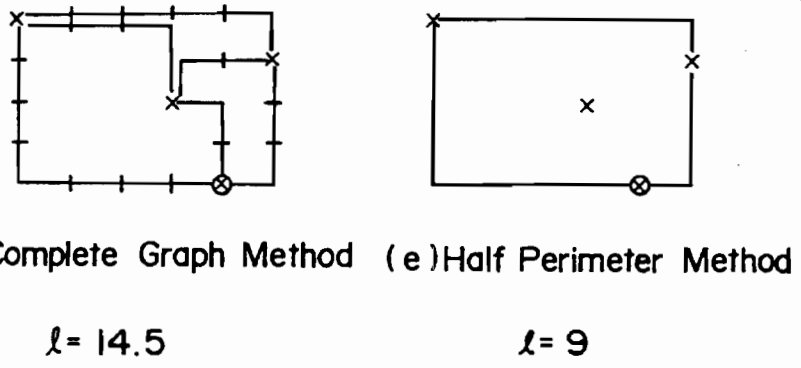


Fig. 13 Simple Routing Length.

Figure 14 shows a comparison between Minimum Spanning Tree and Complete Graph Method. The comparison between Minimum Spanning Tree and Half Perimeter Method was made by calculating the total routing length of all signal nets in a logic circuit. Strong correlation exists between the two values. The stronger one is between Minimum Spanning Tree and Half Perimeter Method. Therfore, as far as the routing length is concerned, but not the routing pattern, we can adopt Half Perimeter Method because of its easier calculation.

4.3 Placement Goals

In the placement stage, locations of individual blocks are decided on a chip to satisfy a number of constraints in actual cases.

- The routing length of a signal net must be kept within tolerable bounds.
- The heat dissipation or power dissipation level has to be preserved.
- Signal cross-talk must be eliminated.

These constraints are interpreted in the placement stage as follows.

(1) A subset of blocks must be placed within some distance from each other.
(2) Some blocks must be placed in fixed positions on a chip.
(3) Some blocks must be placed in the next position for other specified blocks.

Under the above constraints, a good placement solution has to be obtained. We really don't know whether one placement solution is good or not without trying the actual routing. It takes a considerably large amount of computation time to try the actual routing for only one placement result in the real large-scale problems. In the process of obtaining a better placement solution through heuristic procedures, a large number of placement solutions will appear for the evaluation. If every placement solution is evaluated by the actual routing, the computation time would be tremendously long.

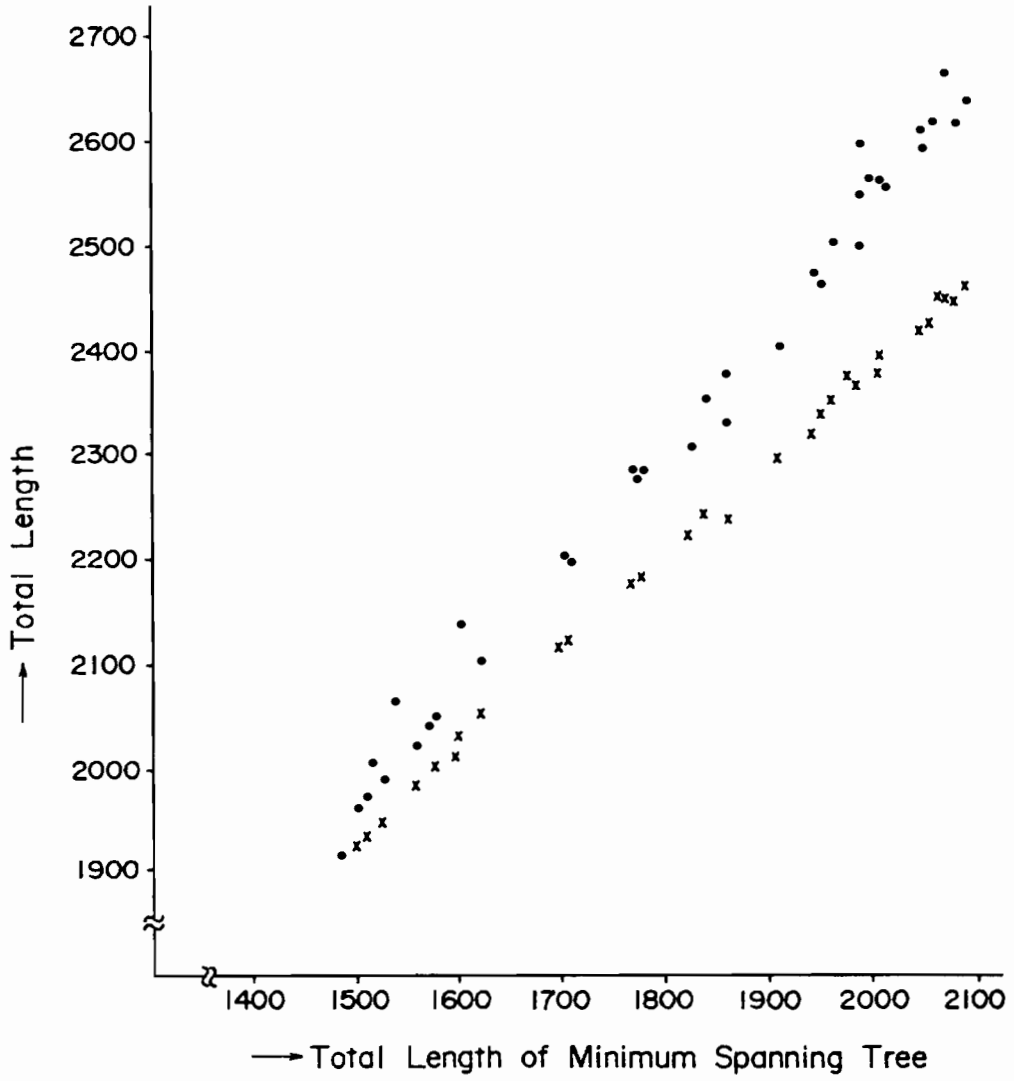


Fig. 14 Routing Length Comparisons.
x : Half Perimeter Method
• : Complete Graph Method

Therefore, the placement and routing problems are solved separately, not simultaneously. A simplified objective is introduced at the placement stage, which is easy to calculate and reflects the actual routing quite well. We will try to optimize a simplified objective, which hopefully leads to the true goal. Three simplified objectives have been proposed until now and are discussed in the followings.

4.3.1 Total Routing Length

The total routing length over all signal nets, T(P), for a placement P, is given by

$$T(P) = \sum_{i,j} W[i,j] \cdot d[P(i), P(j)], \tag{9}$$

if every signal net is common to only two blocks. Here, i and j designate block terminals, w[i, j] is a measure of the nets between the two block terminals i and j, and d[p(i), p(j)] is the distance between i and j. We want to find a placement which minimizes T(P) for all possible placements. If there exists a signal net with more than two blocks, distance, d, is replaced by the length of either one of trees described in Section 4.2.

The total routing length over all signal nets is the area size for the routing itself on a chip. If the routing area involves smaller size, the chip size becomes smaller for a dedicated custom LSI or the routability becomes higher for a fixed size chip LSI like gate-array LSI. The total routing length criterion neglects the interaction between signal nets, since the length is calculated independently for each signal net. This simplified calculation does not reflect the real routing area, but only realizes a rough approximation.

4.3.2 Maximum Cut Line

Maximum cut line, X(P) and Y(P), for placement P, is given by

$$X(P) = \max_i C_p(x_i) \tag{10}$$

$$Y(P) = \max_i C_p(y_i) \tag{11}$$

Here, Cp(xi) and Cp(yi) denote the number of signal nets which cross the vertical line X=xi and horizontal line Y=yi, respectively. A considerable number of vertical or horizontal lines is used to calculate the maximum cut line. We want to find a placement which minimizes X(P) or Y(P) for all possible placements. For a gate-array LSI, a line passing through boundary between cells is taken up as vertical or horizontal line. If we set the vertical or horizontal line for every grid (unit length), the total routing length is calculated as

$$T(P) = \sum_i C_p(x_i) + \sum_i C_p(y_i), \tag{12}$$

which takes the total sum for every component instead of taking the maximum value only.

The difference value between the maximum cut line and the routing capacity of a chip represents the routability of signal nets. A greater positive value indicates a higher routability. If the maximum cut line is over the routing capacity on a fixed size chip, no routing procedure can achieve complete connection.

4.3.3 Maximum Density

Maximum density, D(P) for a placement P, is given by

$$D(P) = \max_i d_p(e_i), \tag{13}$$

$$\text{where} \quad d_p(e_i) = \frac{f_p(i)}{C_p(ei)}$$

Here, $Cp(e_i)$ is the channel capacity for edge e_i and $fp(e_i)$ indicates the number of signal nets assigned to edge e_i .

In the case of the maximum density goal, the whole chip is divided into a rectangular array of blocks, called a portion, each of which contains some wiring grid lines and circuit pins. Global routing determines, for each signal net, a path of portions which the net will be routed through. Global routes for all signal nets are assigned in such a way as not to exceed the number of available grid lines, $Cp(e_i)$, or to minimize the value $fp(e_i)/cp(e_i)$ for all edges. The value $fp(e_i)$ is the number of signal nets whose routes actually pass through edge ei. This problem is similar to a multi-commodity flow problem, which is considered to be a hard combinatorial one, and the real optimum solution is not obtained within a reasonal computation time. Hence, heuristic algorithms are employed. In this situation, we calculate $dp(e_i)$ for each edge, called density.

The maximum density criterion reflects the routability of signal nets more precisely than the maximum cut line criterion. More detailed information regarding the routability inside the chip can be obtained by calculating the density. If the maximum density is greater than one for a placement, complete routing is almost impossible for the placement.

4.3.4 Comparison between goals

The above three objectives lead to different solutions for each other. In many cases, the optimum solution for one objective guarantees an optimum solution for another objective. A solution with the least total routing length quite often satisfies minimizing the maximum cut line or minimizing the maximum density. We don't know exactly how often it happens. We illustrlate solutions with a seven-block problem. The following are 5 signal nets.

$$S1 = \{A1, B1\}$$

$$S2 = \{A2, B2\}$$

$$S3 = \{A3, C1\}$$

$$S4 = \{C2, D1\}$$

$$S5 = \{E1, F1, G1\}$$

Let the chip have 3 rows and 3 columns. The distance between two adjacent cells is defined as, either vertical or horizontal, one unit length. The capacity between two adjacent cells is equal to one unit. The problem is to place all blocks on the cells so that each objective is minimum.

Placement p1, shown in Fig. 15(a), realizes a solution with the minimum total routing length T(p1)=7. However, p1 has the maximum cut line, X(p1)=4 and Y(p1)=2, since

$$X(P1) = Max\{4, 0\}$$

$$Y(P1) = Max\{2, 1\}$$

Also, p1 has maximum density D(p1)=2, since two signal nets have to be assigned to at least one section edge by any routing under this placement,

$$D(P1) = Max\{2, 0, 1, 0, \cdots, \} .$$

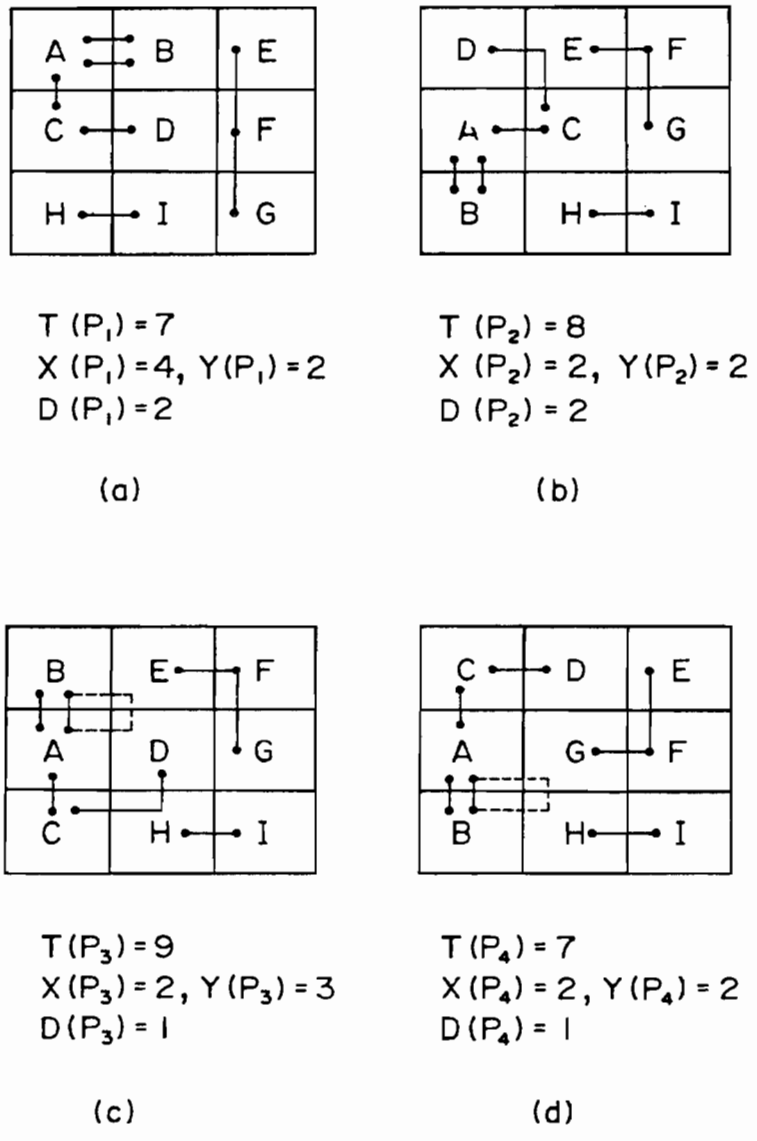


Fig. 15 Placement Results and Goals.

For placement P2, each value is calculated and shown in Fig. 15(b). Placement P2 realizes a solution with minimum vertical and horizontal maximum cut line X(P2)=2, Y(P2)=2. However, the total routing length is longer than that for P1.

Although placement P3 does not realize a solution with either the minimum total length or the minimum maximum cut line, it achieved the minimum maximum density, D(P3)=1, shown in Fig. 15(C). One of the signal nets between block A and B can be routed as A→D→E→B, and, at most, one net is assigned to each edge.

Each placement P1, P2 and P3 realize the optimum solution for one of the objectives, but not the optimum solution for the remaining two objectives. Therefore, we cannot say that the optimum solution for one objective does not guarantee the optimum solution for the other objectives. However, placement P4, shown in Fig. 15(d) realizes the optimum solution for three different objectives.

The problem of finding an actual optimum solution for even one of the objectives is considered to be hard combinatorial problem, which requires exponential order computation time, when the problem size become larger. Therefore, algorithms based on heuristic rationale have been employed. The problem of finding the real optimum solution for all objectives is a much harder combinatorial problem, and sometimes such a solution does not exist.

In the followings, we will discuss heuristic algorithms for either one of objectives, which have been proposed up to now.

4.4 Minimum Length Algorithms

There are, in general, two types of heuristic methods for this kind of problem. One is a constructive method, which obtains a solution using heuristic rules, often in sequential manner. The other one is an iteractive improvement method, which improves a solution by means of local transformations. The algorithms described here are of these two methods, and published in [16]. The constructive ones here are rather simple algorithms and more complex ones are described in Sec. 4.8.

Consider a two-dimensional chip on which blocks are to be placed. The chip is characterized in terms of a finite array of cells. A matrix location, or cell may be represented by a point in an x-y coordinate system. Blocks are the entities which are to be assigned to cells on the chip. It is required that one block can occupy one and only one cell, and on each cell not more than one block is allowed.

Blocks contain pins for connection by phyical wires to form signal nets. In this study, the pins on the blocks are ignored and the distance is measured from the center of the block. Hence, a signal net becomes a subset of blocks, and a signal net specification defines the connection of all blocks on a specific chip.

Let the chip have m rows nd n columns. It may be assumed that the number of blocks is equal to m×n without loss of generality, because dummy blocks, which are not connected, can always be introduced. The distance between two adjacent cells is defined as, either vertical or horizontal, one unit length.

The routing length of a signal net is defined here as half perimeter of the smallest rectangle, which encloses the blocks in the signal set. The placement problem is now defined in the following.

Given a set of blocks with signal nets defined on subsets of these blocks and a set of cells, place all the blocks on the cells so that the total routing length over all signal nets is minimum.

4.4.1 Median of a block

Preceding a placement algorithm, it is necessary to introduce a concept, called median of a block, and present an algorithm to find it, since the present placement algorithm depends on it. Let us consider a chip on which every block is placed. Pick one block, denote it by M. Move only block M on the board, while the other blocks remain fixed. The routing length of a signal net does not change, as long as the signal net is not connected to block M. Therefore, consider the signal net connected to block M only and the sum of the routing length of these signal nets. This value is referred to as the routing lenght associated with block M.

Now, define the median of block M. Block M may be placed on m×n different positions. The block M median is defined as a position where the routing length associated with block M is minimum. Next, sort all the routing lengths associated with block M with respect to the block M position in ascending order. In this order, choose ε elements from the minimum one. The set of these ε position is defined as the ε-neighborhood for block M median.

Now consider how to find a median of a block. Let i (i = 1, 2, · · · , r) designate a signal net which is connected to block M. For each signal net i, consider the smallest rectangle which encloses the block in the signal net. Here, block M is excluded from the signal net when forming the rectangle. Let us denote the rectangle by I_i and its figure by parameters (X_i^a, Y_i^a) and (X_i^b, Y_i^b), where X_i^a and X_i^b are the minimum and maximum values in the x-direction on the rectangle, respectively. The same definitions are pertinent for Y_i^a and Y_i^b in the y-direction.

The routing length associated with block M, which is requied to place block M in position (x, y), is given by

$$F(x,y) = \sum_{i=1}^{r} (f_i(x) + f_i(y)) \tag{14}$$

where

$$f_i(x) = \begin{cases} x_i^a - x, & x < x_i^a \\ 0, & x_i^a \leqq x \leqq x_i^b \\ x - x_i^b, & x > x_i^b \end{cases} \tag{15}$$

$$f_i(y) = \begin{cases} y_i^a - y, & y < y_i^a \\ 0, & y_i^a \leqq y \leqq y_i^b \\ y - y_i^b, & y > y_i^b \end{cases} \tag{16}$$

The problem is to find a pertinent position (x, y) on the chip, such that F(x, y) is minimized. Since the function F(x, y) has a separable form with respect to variables x and y, F(x, y) can be calculated independently from each other for x and y. The y-component can be found in the same way as the x-component. Thus only the x-component will be discussed in the follwoing. Equation (15) is transformed as

$$f_i(x) = \tfrac{1}{2}\{|x - x_i^a| + |x - x_i^b| - (x_i^b - x_i^a)\}. \tag{17}$$

The problem is reduced to finding a position where

$$\sum_{i=1}^{r} (|x - x_i^a| + |x - x_i^b|)$$

is minimum, since x_i^b-x_i^a is a constant value. The value $|x-x_i^a|+|x-x_i^b|$ indicates the sum of the distances from x to x_i^a and x to x_i^b, thus the problem is to find a point x such that the total sum of the distances from x to each point x_i^a, x_i^b ($i = 1, 2, \cdots, r$) is minimum.

In general, it is necessary to solve the following problem. There are a_i points on position i ($i = 1, 2, \cdots, n$) along the line. Find a position x on the line such that

$$\sum_{i=1}^{r} a_i |x - i|$$

is minimum. This problem is a particular case of finding an absolute median of a graph presented in [17]. The present problem, treating only a linear tree instead of a general graph, can be easily solved by using the following theorem.

Theorem I: Point q is the median if

$$\sum_{i=1}^{q-1} a_i < \frac{N}{2} < \sum_{i=1}^{n} a_i \quad \textit{holds,} \tag{18}$$

where

$$N = \sum_{i=1}^{n} a_i \quad .$$

Fact 1

The total distance decreases monotonically from point 1 to a median. From a median to point n, it also increases monotonically. This fact is quite useful when more than one median are interested.

Fact 2

The x-component and y-component of a median can be calculated independently from each other. Each component has the characteristics mentioned in Fact 1. Therefore, the median on the two-dimensional chip can be found easily. When interest is in finding the kth minimum, instead of the first minimum, it is possible to use the efficient algorithm reported in [18]. Thus the ε-neighborhood of a block can be easily obtained.

4.4.2 Constructive Algorithms

This method selects blocks, one at a time, based on an evaluation function which measures signal net connectivity to blocks already or not yet selected, and then decides which cell the selected blocks will be placed on. Once a block is fixed into a position, it is not moved. The conventionl evaluation function, called IOC in Sec. 2.3, is adopted.

$$IOC = I - O$$

or

$$IOC = I \quad .$$

Here, I is the sum of signal net connectivity to all placed blocks for each unplaced block. And O is the sum of signal net connectivity to all unplaced blocks for each placed block. The block with the first or the second highest IOC is the logical candidate to be selected. Each of them is seleced at random as the blocks to be put in place. The selected block is placed on the cell which yields the minimum total routing length among available cells. All available cells need not be examined here. Only a small part of them is examined, by calculating the ε-neighborhood of the mediam.

4.4.3 Iterative Improvement Algorithms

The purpose of this algorithm is to improve the placement by applying small local changes, such as pair-wise interchange of blocks. There is a large number of trials and it is essential that the total routing length is calculated on an incremental basis. There are many ways to interchange blocks efficiently, which are described in [14, 15]. We will just refer the names.

PI: Pairwise-Interchange
NI: Neighborhood-Interchange
FDI: Force-Directed-Interchange
FDR: Force Directed-Relaxation
FDPR: Force-Directed-Pairwise-Relaxation.

In this section, we will describe the details of GFDR (Generalized-Force-Directed-Relaxation) method. This method is considered to be a general one among relaxation methods and a most efficient method.

Let S be the set of all feasible solutions and let x be a feasible solution, $x \in S$. Consider the neighborhood of x, denoted by X(x), which is a subset of S. In the first step, x is set to a feasible solution and a search is made in X(x) for a better solution x′ to replace x. This process, which is referred to hereafter as a local transformation, is repeated until no such x′ can be found. A solution is said to be locally optimum if x is better than any other elements of X(x).

A lot of definitions may be considered for the neighborhood of a solution. In [19], the set of solutions transformable from x by exchanging not more than λ elements is regarded as the neighborhood of x. A solution x is said to be λ-optimum, if x is better than any other solutions in the neighborhood in this sense.

Although the λ-optimum solution gets better as λ increases, the computation time easily goes beyond the acceptable limit, when an exhaustive search is performed for large λ. The following method does not examine all the elements in the neighborhood, nor does it guarantee a λ-optimum solution. However, it is very efficient in the sense that it can be applied for a large value of λ with limited searches in the neighborhood.

The present search procedure is illustrated along with the search tree shown in Fig. 16, where each node represents a block and each edge represents a trial transformation. The root node of the tree A is a block chosen to initiate the trial interchange, it is referred to as the primary block. A path connecting node A and one of the other nodes defines a possible interchange. For example, the path A→B→E→O refers to the trial interchange of four blocks, as shown in Fig. 17. Here, block A is placed on the cell of B, B is placed on E, E on O, and O on A, in a round robin sequence. Although this transformaiton is a quadruple interchange, it includes a pairwise interchange as a special case, i. e., paths A→B, A→C, and A→D, as shown in Fig. 18. Value λ indicates the number of blocks to be interchanged.

The search tree is examined as follows. In this example, ε is fixed as 3. First, block A is interchanged with either one of the blocks on trial in the ε-neighborhood of A median ($\lambda=2$). The ε-neighborhood blocks are B, C, and D, thus pairwise interchanges between A and B, A and C, and A and D are performed (see Fig. 18). The trial interchange is accepted if it results in the reduction of the total routing length. If more than one reduction occurs in these transformations, the interchange with the greatest reduction is selected for

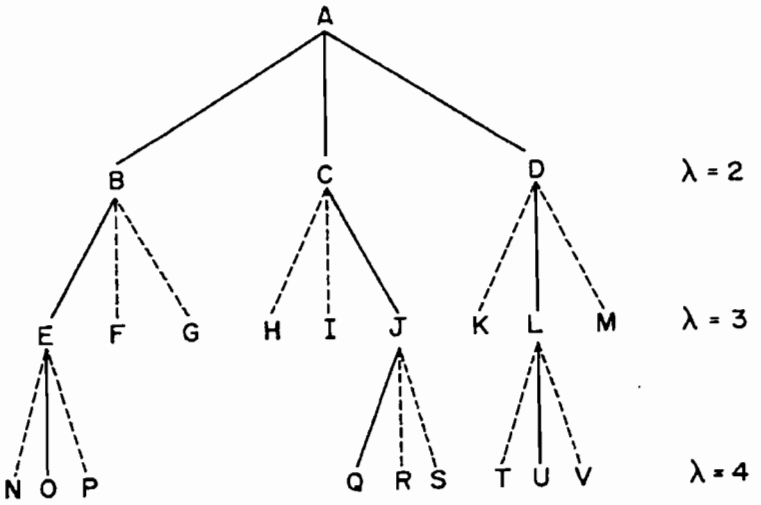


Fig. 16 Search Tree

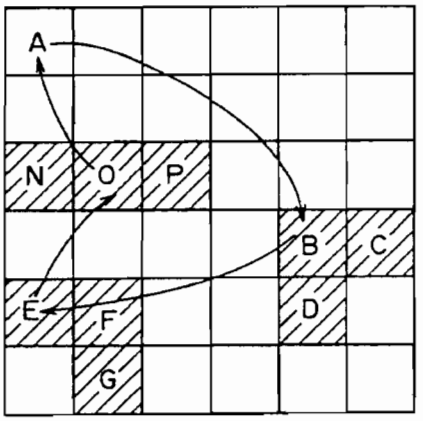


Fig. 17 Trial Interchange of Blocks (=4).

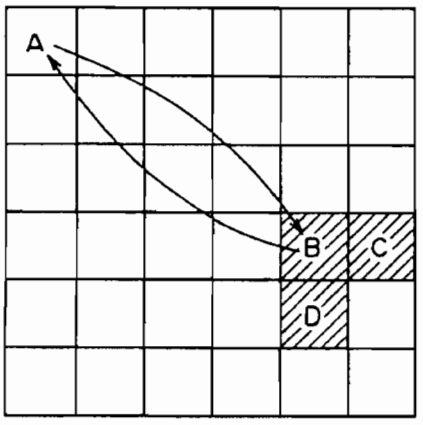


Fig. 18 Trial Interchange of Blocks (λ =2).

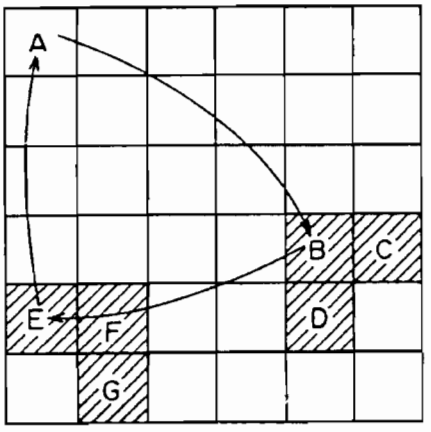


Fig. 19 Trial Interchange of Blocks (λ =3).

acceptance. If no interchange contributes to reducing the total routing length, the next step ($\lambda=3$) is initiated.

Block A is placed on the cell of B. Then the median of B and its ε-neighborhood are calculated. In this case, the ε-neighbohood blocks are E, F, and G. Thus interchanges A→B→E, A→B→F, and A→B→C are tried (see Fig. 19).

These trial interchanges are accepted if one of them results in the reduction of the total routing length. Otherwise, consider the three interchanges of paths A→B→E, A→B→F, and A→B→G, and choose the best one (least total routing length) for the later tree search. Here, A→B→E is chosen, and A→B→F and A→B→G are omitted.

The solid lines in the tree search shown in Fig 16 indicate which searches are to be continued. Broken lines show the searches which are to be terminated. Terefore, no more search efforts are made along paths A→B→F and A→B→G. There is only one solid line under any node, except for root node A. Triple interchanges are performed for the other ε-neighborhood blocks, C and D. of root node A. Tree search will be continued following J or L, whereas no search will be accomplished through H, I, K, and M. The tree search is continued, i.e., a path from node A is extended as long as λ is no greater than λ^*, which is given as a parameter.

Each selection of a primary block is identified with an interchange cycle. Cycles are iterated until there is no reduction in the total routing length

The GFDR method is different from the FDPR method in [15] which tries to interchange only a pair of blocks. The GFDR, on the other hand, tries interchanges of more than two blocks at the same time. If the interchanges are performed randomly in the GFDR, like in the PI method of [5], the gain will not compensate for the comparative great increase in computation time. The GFDR method examines and performs trial interchanges for subsets of blocks which have a large possibility for improvement. This limited trial interchanges enable us to find a good locally optimum solution quickly.

Values ε and λ^* greatly affect the computation time and the total routing length. For a greater value of ε and λ^*, a better solution is obtained at the expense of computation time. In order to find a better solution within a given amount of computaiton time, the key problem is to determine how to set values ε and λ^*. In the following section, several experimental results are shown to point to the best values of ε and λ^*.

4.4.4 Efficiencies of Algorithms

In order to check and compare the results of the present algorithm with others, the algorithm was programmed and tested for five examples in a real problem. The program was written in FORTRAN and run on NEC ACOS-77/700.

Example logic graphs were obtained from [20]. Statistics for the example logic graphs and grid sizes are shown in Table I.

Experiment 1

The placement problem treated here has many locally optimum solutions. Thus different initial solutions lead to different solutions followed by an iterative improvement. It has been debated in the literatures whether it is better to use a random start or to use a constructive-initial solution, followed by an iterative-improvement. Experimental results in [15] and [21] showed that the constructive-initial start approach is superior to the random start in both solution value and computation time. Taking this fact into account, this constructive method generates good solutions randomly as initial solutions. In order to know the relation between the initial solution and its locally optimum solution, Example 5 was provided as a test with $\varepsilon=4$ and $\lambda^*=4$, and 25 different initial solutions were generated by SORG and improved by GFDR.

Table I Statistics for example graphs.

	Example 1	Example 2	Example 3	Example 4	Example 5
# blocks	67	108	116	136	151
# signal blocks	132	277	329	432	419
Av. # signal nets per blocks	6.32	6.41	7.30	7.86	5.98
Av. # blocks per signal net	3.43	2.78	2.66	2.73	2.35
Placement grid size	5×15	8×15	8×15	10×15	11×15

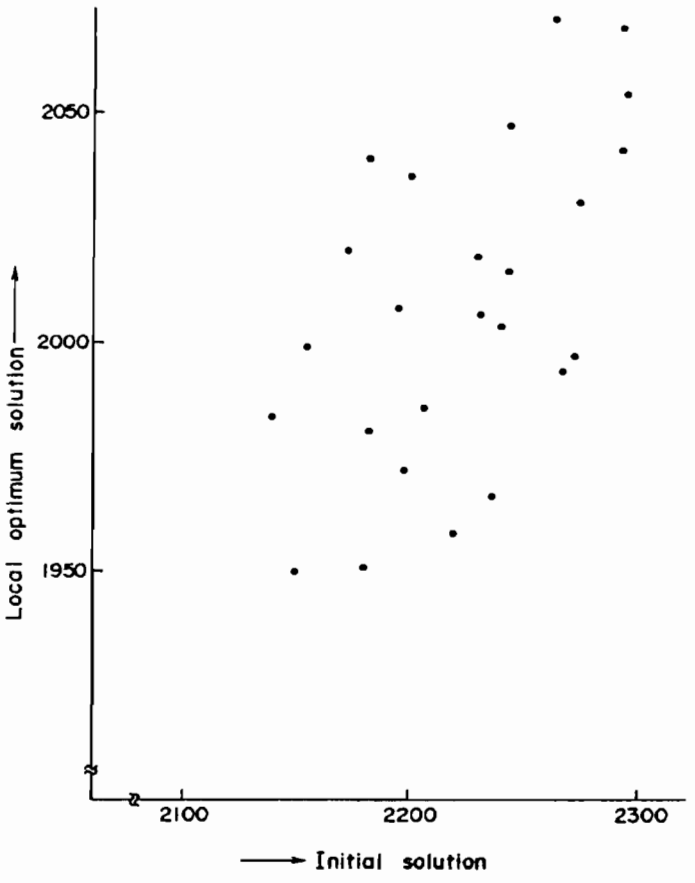


Fig. 20 Relation between Initial Solutions and Locally Optimum solutions.

Figure 20 shows the relation between initial solutions and their locally optimum solutions. A better initial solution does not always result in a better locally optimum solution. This result does not recommend generating many good random starts and following only the best one with iterative improvement. But it recommends repeating random generation of an initial solution and its improvement to obtain a set of localy optimum solutions. The best one among them is chosen as a final solution.

Experimnt 2

A simple transformation, such as pairwise interchange with smaller ε in general, does not yield better locally optimum solutions than more complicated ones with larger ε or λ^*. However, from the computational complexity point of view, the latter one takes much more time than the former one. In order to explore the influence of value ε or λ^* on computation time and the solution, the program was run by changing the value of ε or λ^*.

In Fig. 21, total routing length versus computation time curves are plotted for five values of ε operating on Example 5. Here, λ^* is fixed as 4. Each mark on the curves represents one cycle of GFDR method. The curve with smaller ε results in steeper descent, whereas it does not converge to a better solution. On the other hand, the curve with larger ε converges to a better solutions at the expense of computation time. A fairly small value of ε, i.e., $\varepsilon=4$ or 5 is sufficient to lead to good solutions.

In Fig. 22, total routing length versus computation time curves are plotted for five values of λ^* operating also on Example 5, where $\varepsilon=4$. The same discussion can be made for the value of λ^*, as for ε. The value $\lambda^*=3$ or 4 seems to be enough to have good solutions.

The aim of the overall scheme is to generate as many locally optimum solutions as possible within some time interval. Then, the best one among them is chosen. Let P be the probability that the cost of the locally optimum solution is less than V and let T_0 be the running time to produce the local optimum solution. Then, the best locally optimum solution produced within time interval T has a probability

$$P = 1-(1-P)^{T/T_0} \tag{19}$$

of being better than V.

Random starts were repeated as many times as possible within T=30 min.for a value of ε. Here, Example 5 was provided as a test and λ^* was fixed as 4. Let P'_ε be the probability of (19) associated with ε, and calculated P'_ε of being less than 2000. The experimental results show the $P'_2=0.68$, $P'_3=0.71$, $P'_4=0.95$, $P'_5=0.92$, $P'_8=0.70$. This procedure performs best at $\varepsilon=4$.

The same analysis was done for finding the value of λ^*. Let P'_{λ^*} be the probability of (19) associated with λ^* and calculated P'_{λ^*} of being less than 2000. Here, ε was set to 4. We have the results that $P'_2{}^*=0.74$, $P'_3{}^*=0.93$, $P'_4{}^*=0.98$, $P'_5{}^*=0.89$, $P'_8{}^*=0.75$. The best value was $\lambda^*=4$. The same tendency was observed for the other four examples.

Note 1

GFDR method is considered to be a general relaxation one. The algorithm with $\lambda^*=2$ and $\varepsilon=\infty$ corresponds to PI (Pairwise Interchange) method, and the algorithm with $\lambda^*=$ and $\varepsilon=1$ corresponds to FDR (Force-Directed-Relaxation) method.

Note 2

The algorithm with $\lambda^*=2$ did not result in good solutions in the meaning of both the appropriateness for solution value and computation time. This algorithm is called as the FDPR method in [15] and considered to be best among existing algorithms.

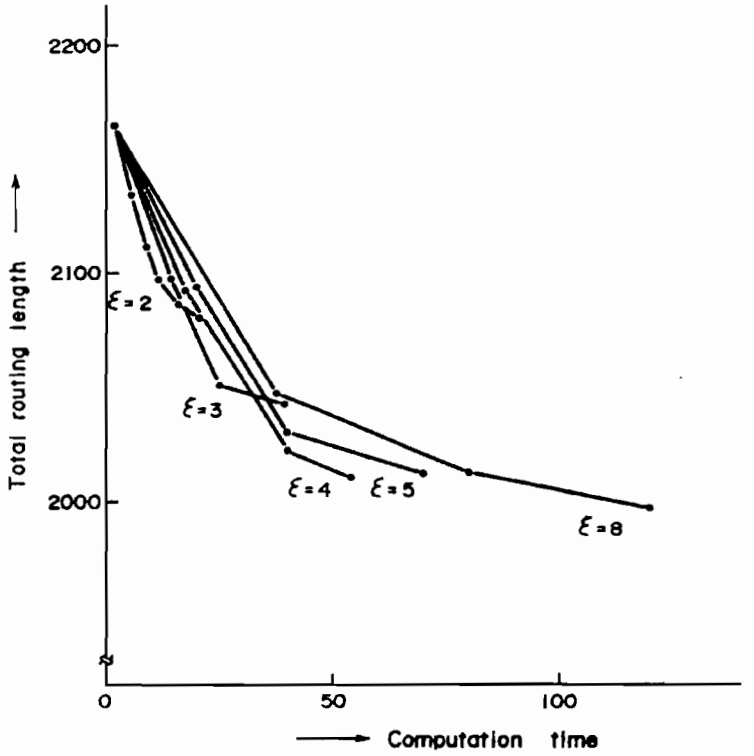


Fig. 21 Total Routing Length versus Computation Time Curve for Example 5, varing the Value of ε .

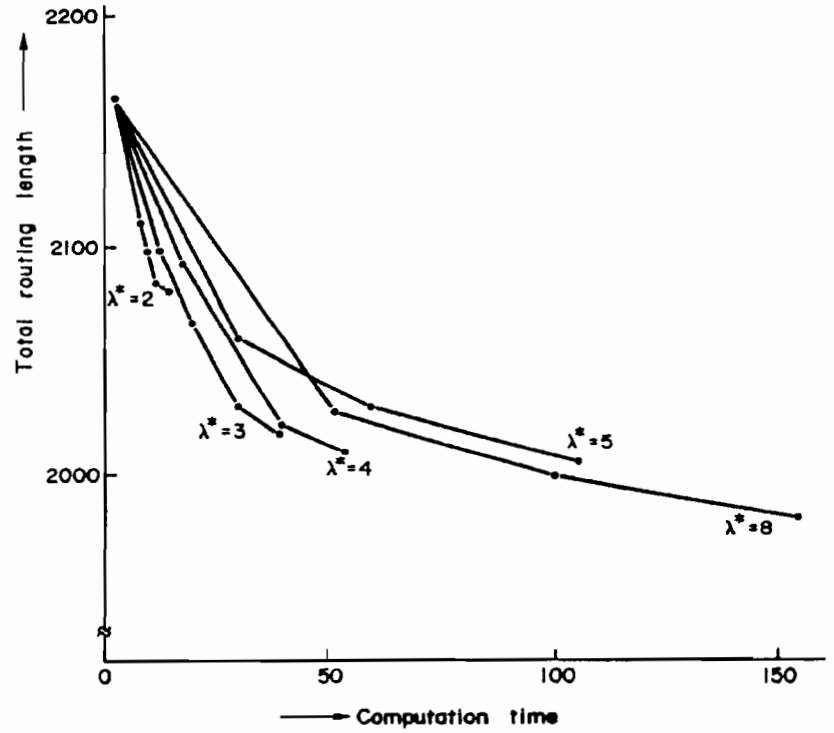


Fig. 22 Total Routing Length versus Computation time Curve for Example 5, varing the Value of λ^* .

Note. 3

The algorithm with larger ε, did not result in good solutions either. The method with the testing of all possible interchanges is exactly equal to the proposed algorithm with the largest value of ε. In this sense, the testing of all possible interchanges is said to be inferior to limiting the number of possible exchanges proposed here.

Experiment 3

In order to explore the influence of the number of blocks on the computation process, five examples were examined.

The graph in Fig. 23 shows computation time versus number of block for various values of ε. The computation time increases quite rapidly for $\varepsilon>6$, when the number of blocks increases. Therfore, it seems infeasible to set $\varepsilon>6$ for large scale problems.

The graph in Fig. 24 also shows computation time versus number of blocks, while varying the value of λ^*. From a practical point of view it may not be reasonable to set $\lambda^*>5$ for large scale problems.

The computation time to obtain a locally optimum solution with $\varepsilon=4$ and $\lambda^*=4$ was linearly proportional to the number of blocks.

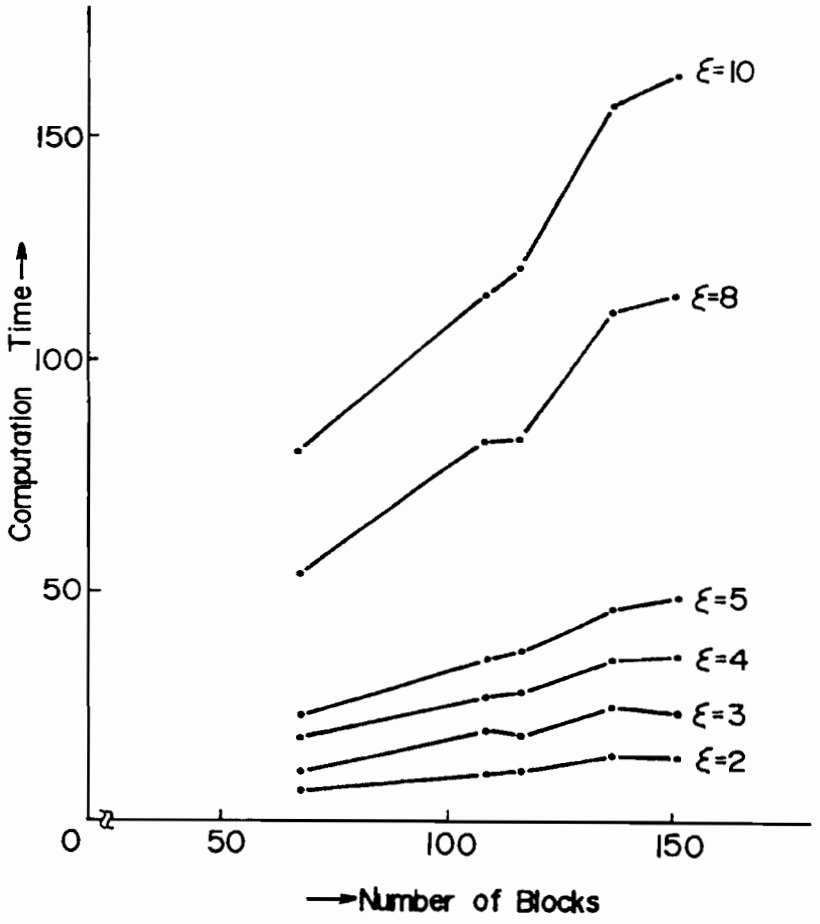


Fig. 23 Computation Time versus Number of Blocks, varing the Value of ε.

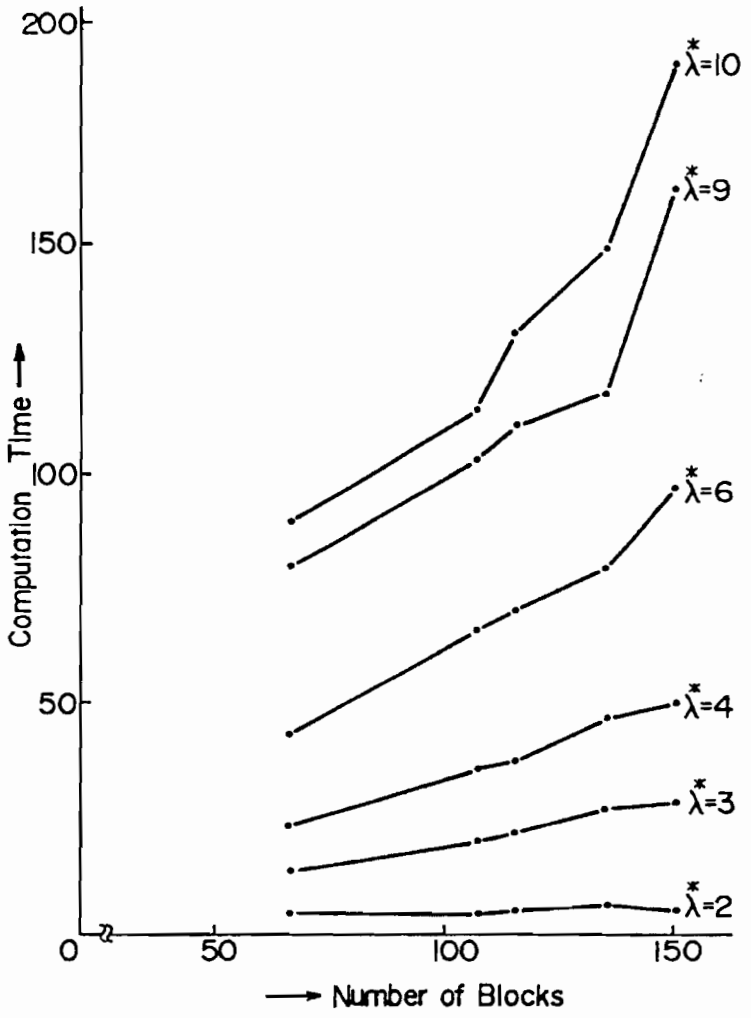


Fig. 24 Computation Time versus Number of Blocks, varing the Value of λ^*.

4.5 Minimum Cut Algorithms

The original idea for this algorithm was given by Breuer[22].

Let C={C1, C2, · · · Cr} be a set of cut lines crossing the chip, either horizontally or vertically, and the value of Ci be the total number of signal nets crossing the cut line Ci, denoted by v(Ci). Objective function is to minimize some function F of the values of each cut line. This kind of placement algorithm is called minimum cut algorithm (or Min-cut algorithm).

An ideal objective function for this algorithm is

$$F1 = min\ (MAX\{v(c) \mid c \in C\}) \quad . \tag{20}$$

However, this function is too difficult to satisfy. In general, following function is used:

$$F1 = min\ v(C_r) \mid min\ v(C_{r-1}) \mid \cdots \mid min\ v(C_1) \tag{21}$$

where C1, C2 $\cdots$, Cr is a given set of cut lines, and " | " can be read as "subject to." Here (1, 2, $\cdots$, r) represents an ordered sequence of cut lines. This objective function is highly dependant on the locations of cut lines, as well as chip geometry.

Breuer's algorithm has been developed to handle regularly structured chips, such as gate-array or poly-cell LSIs. However, it is not easy to apply the algorithm to irregularly shaped macro placement. Therefore, modified algorithms have been presented which are suitable to macro placement.

4.5.1 Basic Algorithm[22]

The algorithm is an iteractive improvement method, which starts with an initially given placement result and improves it by a local transformation.

It is assumed that a chip consists of cells and that every block has been assigned in a cell. Consider the chip shown in Fig. 25, where some arbitrary assignment of blocks to cells has occured. An assignment of blocks to cell locations is refered to as a temporary assignment, denoted by T-assignment. Now, line C divides are A into two areas A1 and A2.

The basic problem is to decide on a re-assignment of blocks in A to cells in either A1 or A2, so that the total number of signal nets crossing C is minimized, subject to the constraint that blocks not in A cannot be moved. New assignments of blocks in A1 and A2 are again considered to be T-assignments.

An area A, along with the cells and blocks within A is called a group denoted by g. (In [22], it is called a block. In this paper, it is called a group to avoid conflict.) A block is considered placed when the group it is in has only one cell.

Let G be a set of disjoint groups. Initially, let the entire chip be group g_1, and set $G=\{g_1\}$. The basic algorithm is as follows.

1) Select a sequence for processing the cut lines.
2) Select next cut line C in sequence.
3) Assume C cuts across a subset of groups $G'=\{gi1, gi2, \cdots, git\}$. Re-assign blocks within these groups, such that v(C) is minimized.
4) In a natural way, form two new groups from each of the groups cut by C.
5) If there are no more cut lines to process, or if every group contains at most one block then stop, or else return to 2).

The sequence in which cut lines are to be processed can be either fixed or adaptive.

Fig. 26 shows an example of this algorithm along with five cut lines which are processed in a sequence C1, C2, C3, C4, and C5. Starting with the initial group g_1, consisting of the entire chip and all blocks, g_1 is processed with respect to C1, hence minimizing v(C1). Group g_1 is now divided into two new groups, denoted by g_1 and g_2. Note that once a block has been assigned to the left (the right) of C1, the block can never be moved to the other side of C1, no matter where subsequent cut lines occur. Now g_1 and g_2 are processed (simultaneously) with respect to C2, producing the four groups in Fig. 26(d). Next C3, which only intersects g_3 and g_4, is processed. Note that the blocks in g_1 and g_2 are not moved. Therefore, the locations of those blocks need not be known when calculating v(C3). Continuing these process, the resulting 12 groups shown in Fig. 26(g), are obtained.

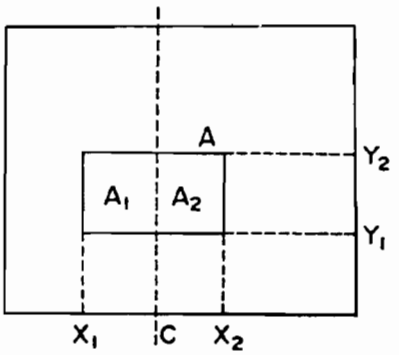


Fig. 25 Chip and Groups

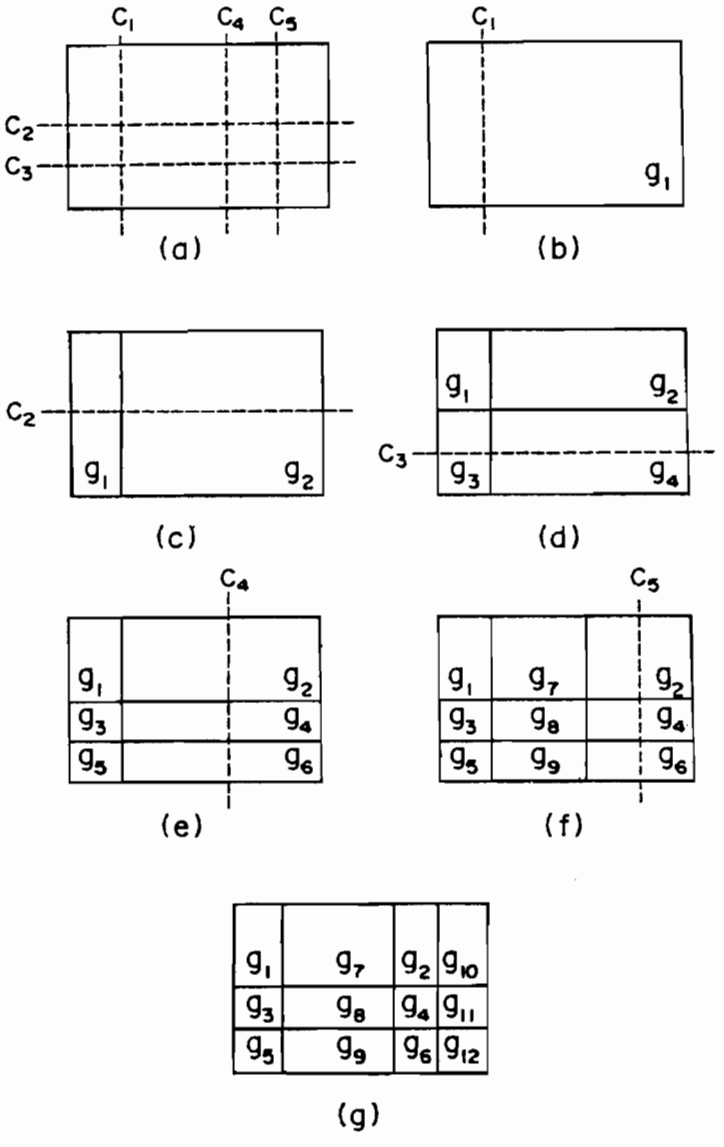


Fig. 26 Min-Cut Algorithm Example

In this algorithm, two key problems exist. One problem is to assign the block in a group g to two groups, g_1 and g_2, such that the number of signal nets crossing across a cut line C is minimized. This problem is a generalization of the following partitioning problems.

Given a graph G having n nodes, partition the set of nodes into two disjoint sets N_1 and N_2 of nodes having n_1 and n_2, respectively, where $n_1+n_2=n$, and such that the number of edges between N_1 and N_2 is minimal.

Heuristic algorithms for this problem have already been discribed in Sec. 2. 3.

Another key problem is to select cut lines and sequence in which cut lines are processed. The selection criteria is usually a function of the chip geometry and predicted routing density. For fixed ordering, it has been found that two types of cutlines are quite effective; they are referered to as a slice cut and a bisection cut. Slice cut C for a group g is a cut line which isolates a fixed number (K) of cells of g to one side of C and the remaining cells to the other side of C. A bisection cut C of group is a cut line which divides the blocks in g into either side of C as evenly as possible.

Two procedures are recommended by Breuer: Quadrature procedure and slice/bisection procedure. They differ in the order in which groups are processed and the type of cut lines employed.

In the quadrature procedure, the original group (entire chip) g_1 is first bisected by a vertical cutline producing two new groups g_1 and g_2. These two groups are then cut by horizontal bisection cut, thus producing four groups, which are next cut by vertical bisection cut. This process is repeated, until every block is placed.

This procedure tends to produce a placement which can be routed with a more uniform density. Therefore, it is suitable for chips having a high routing density in their center.

In slice/bisection procedure (see Fig. 27), initial set of n blocks is divided into a set of K blocks and (n-K) blocks, where $K>0$, such that v(C) is minimized. These K blocks represent the bottom row or slice of chip. This procedure is repeated on the remaing (n-K) blocks, again dividing them into a set of K blocks and (n-2K) blocks. This process is iterated until all blocks have been assigned to rows. The blocks are then assigned to columns via vertical bisection. This technique is best fit for chips where there is a high interconnected density at the external terminals.

These techniques have been widely used in designing gate-array and poly-cell LSIs.

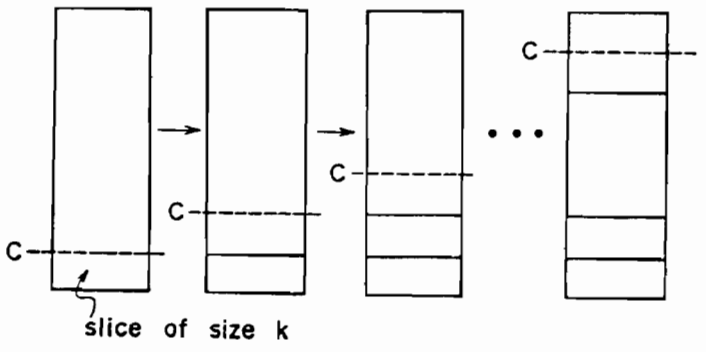


Fig. 27 Slice/Bisection Procedure

4.5.2 Min-cut algorithms for Macro Placement [24]

The basic algorithms described in the previous section are suitable for block placement, where each block has the same size or the same height even though it has different width. On the other hand, macro placement problem has to deal with irregular components which has various kind of shapes or sizes ranging from smaller ones to much larger ones. In addition,exact block location cannot be defined ,since chip shape is obtained after completing placement. Therefore, to apply the min-cut algoritm to macro placement, various block size must be taken into account and particular data structure has to be adopted, which represents the block locations.

Polargraph has been introduced to represent the macro placement by the first author, whose idea was implemented in the routing program ROBIN [23]. Since that time, several macro placement algorithms, based on graph representation, have been proposed [24] [25] [26]. Most of them are based on the min-cut algorithm with some modification or top-down approarch.

Throughout the placement, macro placement is represented by a pair of mutually dual graphs $G_x=(V_x, E_x)$ and $G_y=(V_y, E_y)$. G_x and G_y are planar, acyclic directed graphs containing one source and one sink each; parallel edges are permitted. There is a one to one correspondence between the edges of G_x and G_y. Each pair of edges (e_x^i, e_y^i) represents a rectangle Ai with x-dimension $\ell(e_x^i)$ and y-dimension $\ell(e_y^i)$, where $\ell(e)$ denotes the length associated with edge e. On the other hand, each pair of edges is defined to correspond to also a group gi; the area $\ell(e_x^i) \cdot \ell(e_y^i)$ equals the total area of the macros in the associated gi.

When the algorithm starts, the two graphs contain one edge. This pair of edges represents initial group g with all macros of the circuit. The area covered by group g is assumed to be square.

Therefore,

$$\ell(e_x^1) = \ell(e_y^1) = \sqrt{\sum_i a_x^i \cdot a_y^i}$$

where a_x^i and a_y^i are the dimensions of macro i (Fig. 28 a, b). Group g is now partitioned into two groups, g_1 and g_2, in such a way that the number of nets incident to macros in different groups is minimal and that the difference between total macro areas in the two groups does not exceed a predefined threshold value.

In the graph representation, this step corresponds to a splitting of the edge pair into two new edge pairs, each representing one of the two groups . The lengths of the edges in Gx are adjusted according to the total cell area in the respective group. Note that the two resulting rectangles in Fig. 28(d) have different areas.

In the next step, the partitioning procedure is applied to both of the groups. Now, however, the direction of the cutline is changed and the edge lengths in G_y are adjusted. The resulting situation is shown in Figs. 28(e) and (f). Group partitioning is carried out sequentially, alternating between the two groups, and iterating the process until the overall count of nets cut cannot be further reduced.

The described procedure is applied recursively to the new groups and terminates when each group contains one macro (Figs. 28(g) and (h)). Now, each edge-pair of the two graphs corresponds to one macro and the area $1(e_x^i) \cdot 1(e_y^i)$ of the corresponding rectangle is equal to the area of the respective macro i, whereas the shape of the cell (length to width ratio) is not correctly represented.

It is easy to convert this representation to actual macro dimensions, while maintaining the local relations as derived by the algorithm.

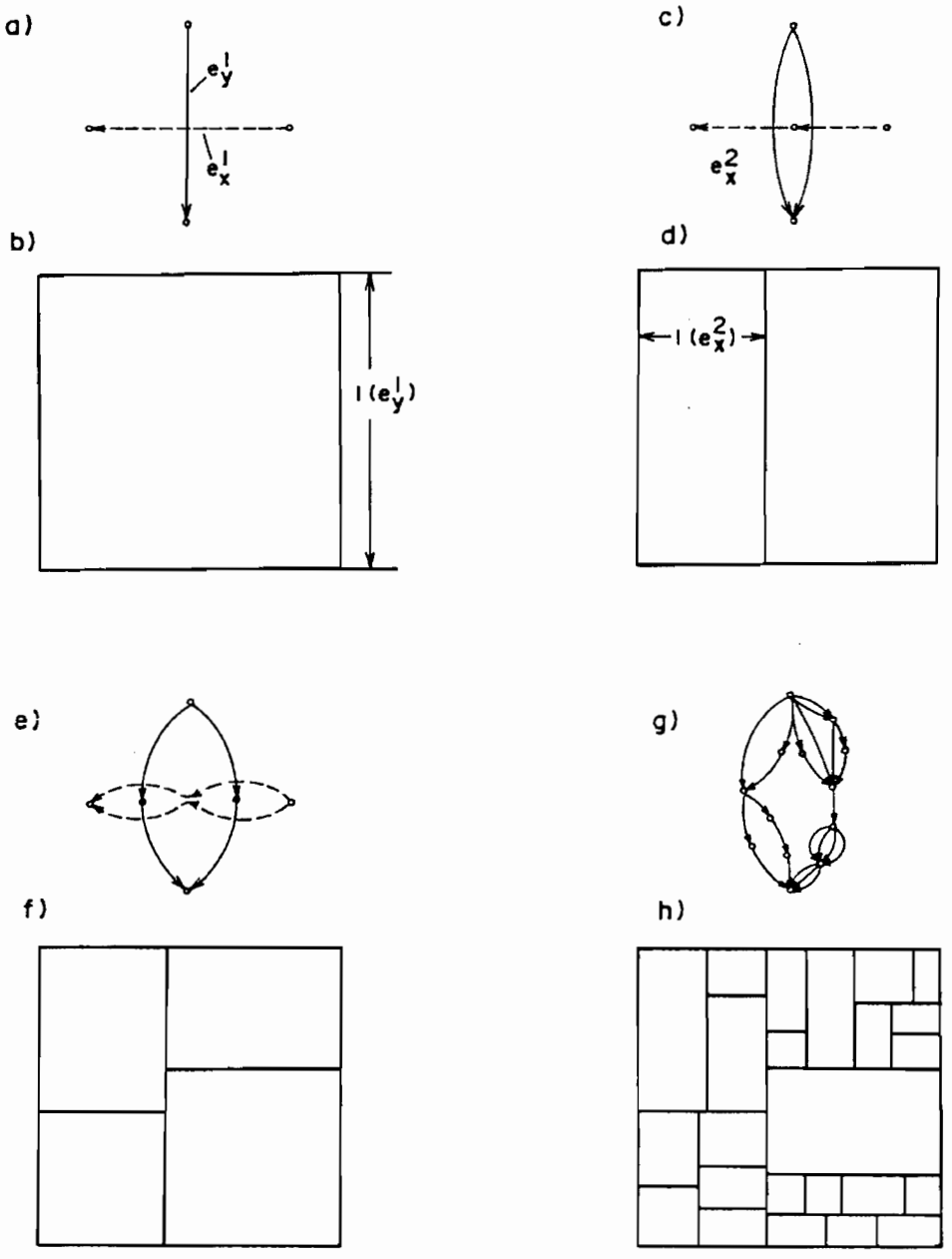


Fig. 28 Macro Placement and Associated Graphs

4.6 Minimum Density Algorithms

In the case of the minimum density goal, the whole chip is divided into a rectangular array of cells, each of which contains some grid lines and circuit pins. Global routing determines, for each signal net, a path of cells which the net will be routed through. The boundary between two cells is called a segment. To each sigment, the number of allowable signal nets, called capacity, is given. The routing path of a signal net is expressed as a sequence of segments. Thus, if the routing path for every signal net is given, the number of signal nets assigned to each segment can be calculated.

In this section, the same problem assumption as in Sec. 4.4 is adopted. We will rewrite Eq. (13) into a new form which is easy for the density calculation.

We now introduce the following symbols (see Fig. 29):

e_{ij}^x: X-directional segment, where $i=1, 2, \cdots m$; $j=1, 2, \cdots, n-1$.

e_{ij}^y: Y-directional segment, where $i=1, 2, \cdots m-1$; $j=1, 2, \cdots, n$.

C_{ij}^x: the number of signal nets segment e_{ij}^x can accomodate

C_{ij}^y: the number of signal nets segment e_{ij}^y can accomodate

X_{ij} : the number of signal nets assigned to segment e_{ij}^x

Y_{ij} : the number of signal nets assigned to segment e_{ij}^y

$f(X_{ij}, r)$: the cost of e_{ij}^x when X_{ij} signal nets are assigned to segment e_{ij}^x, where r is a congestion parameter

$f(Y_{ij}, r)$: the cost of e_{ij}^y when Y_{ij} signal nets are assigned to segment e_{ij}^y, where r is a congestion parameter

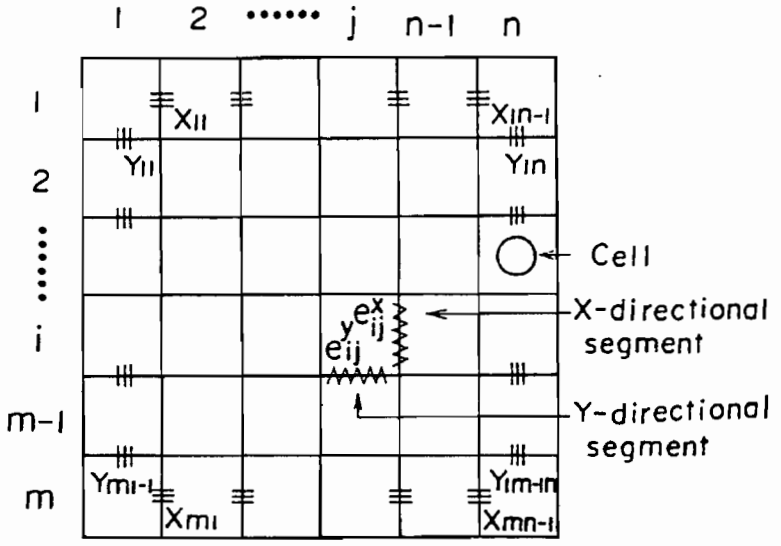


Fig. 29 A chip Structure.

X_{ij}/C_{ij}^{x} and Y_{ij}/C_{ij}^{y} are called segment densities. The cost function is assumed to satisfy the following equation;

$$\lim_{r\to\infty} f(X_{ij}, r) = \begin{cases} 0 & : X_{ij} \leq C_{ij}^{x} \\ \infty & : X_{ij} > C_{ij}^{x} \end{cases} \tag{22}$$

That is, when the segment density is greater than 1 (the number of signal nets assigned exceeds the number of signal nets that can be accomodated), the cost becomes quite high if r becomes large. When the segment density is less than 1 (the number of signal nets assigned is smaller than the number of signal nets that can be accommodated), the cost approaches 0 as r becomes large. There are a number of functions that satisfy Eq. (22). Here we use Eq. (23) (see Fig. 30).

$$f(X_{ij}, r) = \left(\frac{X_{ij}}{C_{ij}}\right)^{r} \tag{23}$$

Cost function $f(Y_{ij}, r)$ can be defined similarly. The cost Fr for the entire segment is defined by the following equation and we take as an objective function the minimization of this cost:

$$\begin{aligned} F_r &= \left(\sum_{i=1}^{m}\sum_{j=1}^{n-1} f(X_{ij}, r) + \sum_{i=1}^{m-1}\sum_{j=1}^{n} f(Y_{ij}, r)\right)^{\frac{1}{r}} \\ &= \left(\sum_{i=1}^{m}\sum_{j=1}^{n-1}\left(\frac{X_{ij}}{C_{ij}}\right)^{r} + \sum_{i=1}^{m-1}\sum_{j=1}^{n}\left(\frac{Y_{ij}}{C_{ij}}\right)^{r}\right)^{\frac{1}{r}} \end{aligned} \tag{24}$$

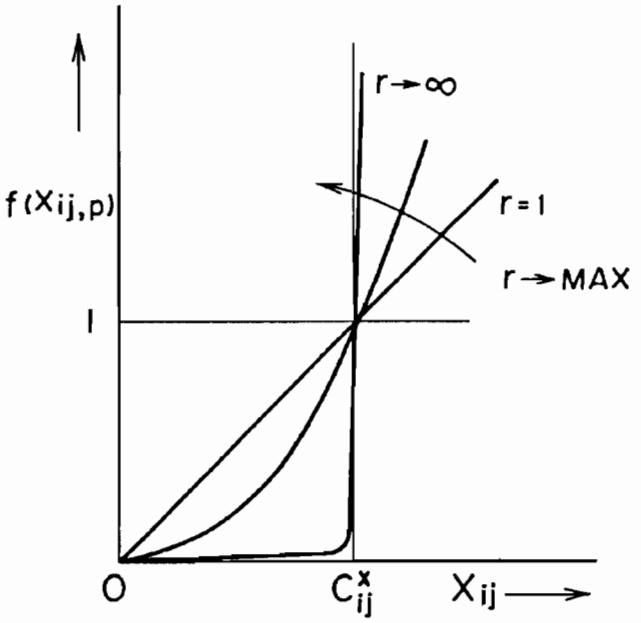


Fig. 30 Cost Function $f(X_{ij}, r)$.

The problem in this section is to place blocks on a chip so that F_r is minimized for a given value of r. By setting an objective function in this manner, we can evaluate various placements on a common ground. For example, if we let $r=1$ and $C_{ij}^x = C_{ij}^y = C$ in Eq. (24), then we have

$$F_1 = \frac{1}{C}\left(\sum_{i=1}^{m} \sum_{j=1}^{n-1} X_{ij} + \sum_{i=1}^{m-1} \sum_{j=1}^{n} Y_{ij} \right) \tag{25}$$

Equation (25) represents the total routing length.

If we let $r \rightarrow \infty$, then

$$F_\infty = \underset{ij}{MAX}\left(\left(\frac{X_{ij}}{C_{ij}^x} \right), \left(\frac{Y_{ij}}{C_{ij}^y} \right) \right) \tag{26}$$

Equation (26) gives the maximum segment density.

4.6.1 Calculation for density

Strictly speaking, the goodness of a placement should be judged by whether or not 100% wiring is possible when an ideal routing is taken. However, this makes the problem very complex. Thus, in practice, we introduce a set of criteria which approximately evaluate placements and isolate the placement problem from the routing problem. [27]

First, we calculate for each segment the number of signal nets which pass it. We make the following assumptions for routing since they produce routings close to actual cases using simpler calculations.

Assumption 1. The routing path of a signal can be determined independently of the routing paths of the other signal nets.

Assumption 2. The routing path of a signal is inside the smallest square containing the blocks connected by the signal net.

Assumption 3. The direction of a routing path is parallel to either the x-axis or the y-axis and it can make at most two turns.

Assumption 4. When more than one routing path is possible for one signal net, each of them has an equal probability of occurrence.

Complex processes and hence considerable computation time are required to consider the effects of signal nets on each other. Hence Assumption 1 is adopted. Also, most of the routings in practice satisfy Assumptions 2 and 3. Assumption 4 means that there are no priority orders among routing paths satisfying Assumptions 2 and 3. If priority needs to be given to a particular routing path, we can assign a higher probability to the routing path.

First we obtain the number of wires for the case of two-point wirings under Assumptions 2 and 3. Let A and B be blocks and let (x_A, y_A), (x_B, y_B) be the cells for them, respectively. There are four possible combinations of their positions, but we consider only the case $x_A \leqq x_B$, $y_A \leqq y_B$. The other three cases can be treated similarly. Let l_{AB} be the number of routing paths connecting A and B. Then

$$l_{AB} = \begin{cases} 1 : x_A = x_B \ \text{or} \ y_A = y_B \\ (x_B - x_A) + (y_B - y_A) : x_A = x_B \ \text{and} \ y_A = y_B \end{cases} \tag{27}$$

Let $l(e_{ij}^x)$ be the number of signal nets passing through segment e_{ij}^x. Then it can be obtained by the following equations:

(1) If $x_A = x_B$, then

$$l(e_{ij}^x) = 0 \tag{28}$$

(2) If $x_A \neq x_B$, and $y_A = y_B$, then

$$l(e_{ij}^x) = \begin{cases} 1 : x_A \leqq i \leqq x_B - 1 \ \text{and} \ j = y_A \\ 0 : \text{otherwise} \end{cases} \tag{27}$$

(3) If $x_A \neq x_B$, and $y_A \neq y_B$, then

$$l(e_{ij}^x) = \begin{cases} x_B - i : x_A \leqq i \leqq x_B - 1 \ \text{and} \ j = y_A \\ i - x_A + 1 : x_A \leqq i \leqq x_B - 1 \ \text{and} \ j = y_B \\ 1 : x_A \leqq i \leqq x_B - 1 \ \text{and} \ y_A < j < y_B \\ 0 : \text{otherwise} \end{cases} \tag{30}$$

The number of signal nets $l(e_{ij}^y)$ passing through e_{ij}^y can be obtained similarly. Let X_{ij} and Y_{ij} be the number of signal nets assigned to segments e_{ij}^x and e_{ij}^y, respectively, for a signal net connecting blocks A and B. Then from assumption 4 we can obtain

$$\left.\begin{aligned} X_{ij} &= l(e_{ij}^x) / l_{AB} \\ Y_{ij} &= l(e_{ij}^y) / l_{AB} \end{aligned}\right\} \tag{31}$$

Figure 31 enumerates routing paths between blocks A and B and $l_{AB} = 5$.

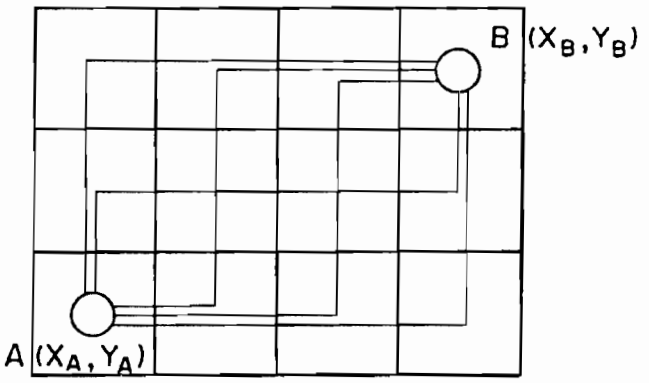


Fig. 31 Routing paths between Block A and B.

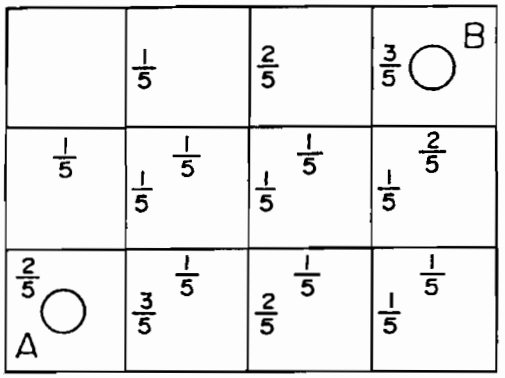


Fig. 32 Number of Signal Nets assigned to each Segment.

Figure 32 shows the number of signal nets that a routing path connecting blocks A and B assigns to each segment. If there are multipoint wirings among given signal nets, they are decomposed into two-point wiring by the method given in section 4.2. Then, the number of signal nets to be assigned to each segment is calculated by Eqs. (27) to (31) and the total number of signal nets for each segment is obtained. This value corresponds to X_{ij} and Y_{ij} and Eq. (24) can now be calculated.

4.6.2 Algorithms and experimental results

Since Eq. (26) is considered to be an extreme case of Eq. (24), GFDR method mentioned in Sec. 4.4 is efficiently applied. We will describe the results obtained in [27]. Here, Example 5 of Table I was provided as a test and $C_{ij}^x = C_{ij}^y = 10$ was used.

(1) The total routing length and the maximum density

We gave a value to the density parameter r, let $\lambda^* = 4$ and $\varepsilon = 4$. As a result, a placement of blocks was obtained and the total routing length and the maximum segment density were calculated for that placement. For the value of r, 1, 2, 4 and 8 were used. The total routing length and the vertical and horizontal maximum segment densities were obtained as shown in Fig. 33. The maximum density becomes smaller as the total routing length becomes longer. It has been found that for r, $2 \leqq r \leqq 4$, the total routing length is not large and neither is the maximum density.

(2) Distribution of segment density

The distribution of values of the segment densities (occurrence frequencies) against the change in the wire congestion parameter r is given in Fig. 34. From the figure we can see that for any value of r, the density exceeds 1 for about half the segments. Hence for $C_{ij}^x = C_{ij}^y = 10$, 100% wiring seems impossible. If we set the values of C_{ij}^x and C_{ij}^y at about 15 (in this case the distribution of segment density is obtained by dividing by 1.5 the values of the horizontal axis of Fig. 34), less than 10% of the segments have density greater than 1.0. Thus we can expect 100% wiring if a good routing algorithm is used. When $r = 1$ or 2, the wire congestion of segments has an average distribution. But it does not give good placements in the sense that the maximum density of segments is not minimized.

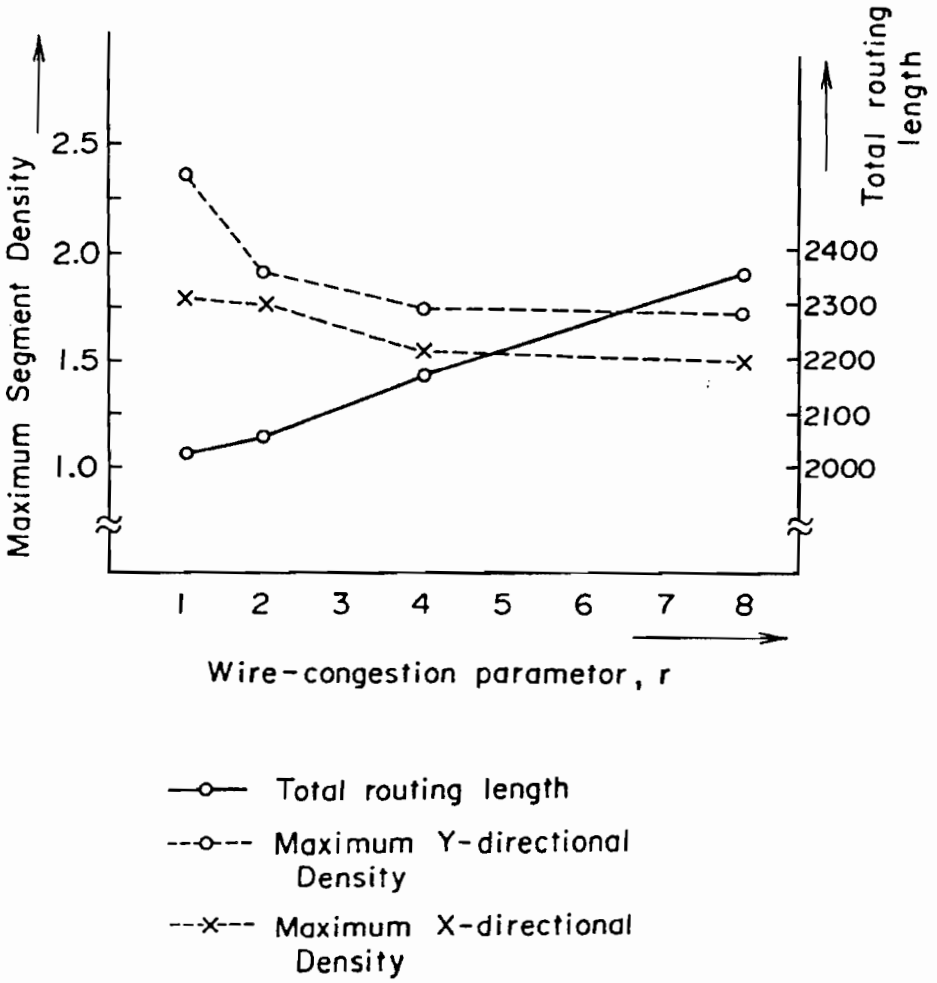


Fig. 33 Relation between the Maximum Segment Density and the Total Routing Length.

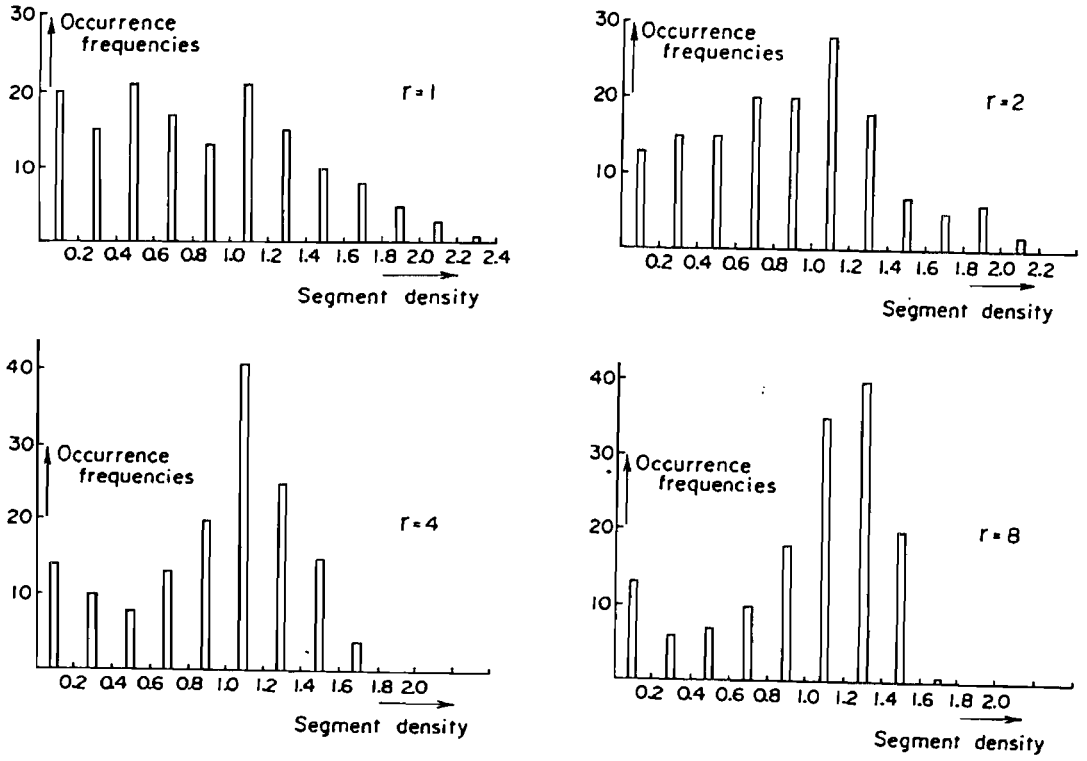


Fig. 34 Occurrence Frequencies of Segment Density with regard to Wire Congestion Parameter r.

5. CONCLUSION

This chapter dealt with partitioning, assigment and placement problems and gave the current status of solving methods. Although those three problems are closely related, they were treated separately in practical cases because of its inherent computational complexities of the total problem. In this chapter, each problem was formulated as a mathematical problem and heuristic solving methods were given. Each problem is considered to be hard combinatorial problem (NP-Complete), and hense algorithms based on heuristic rationales have been employed.

Although these problems have been attached by many researchers in the past 20 years, the results obtained from heuristic algorithms are still not satisfactory, in comparison to an expert designer's result. This is because an expert designer can well understand really complicated problems with numerous constraints, and he can find a good solution by exploiting his knowledge and intuition.

Under this situation, the authors believe that the following approaches are indispensable.

(1) The design problems themselves should be changed or simplified to make the problems easy solvable. One typical example is a gate array LSI designed with sufficiently large chip size for easy routing.

(2) More research should be carried out to investigate various objective functions for the optimum layout. Three different objective functions are presented for the placement problem. However, more detail relations among them should be made clear. Altogether, the research should be forcused toward finding the most appropriate objective function which reflects the complete net connectivity.

(3) If using algorithmic approaches make it essentially impossible to get better results, and real design would require much better result, a new approach should be divised, which would be substantially different from conventional algorithmic approaches. The authors believe that the knowledge based approach incorporated with an algorithmic approach represents the most hoepful from now on.

ACKNOWLEDGMENTS

The author would like to thank Prof. E. S. Kuh of University of California, Berkeley, Prof. T. Ohtsuki of Waseda University, Drs. Y. Kato, M. Naniwada and colleagues of NEC Corporation for their encouragement and useful suggestions.

REFERENCES

[1] Kodres, U. R., "Partitioning and Card Selection," Design Automation of Digital Systems (Ed. by M. A. Breuer), Prentice - Hall (1972).

[2] Lawler, E. L., "Electrical Assemblies With a Minimum Number of Interconnections," IEEE Trans. on Electronic Computers (Correspondence), Vol. EC-11 (February, 1962), pp. 86-88.

[3] Lawler, E. L. et al., "Module Clustering to Minimize Delay in Digital Networks," IEEE Trans. on Computers, Vol. C-18 pp. 47-57 (January, 1969).

[4] Luccio, F. and M. Sami, "On the Decomposition of Networks in Minimally Interconnected Subnetworks." IEEE Trans. on Circuit Theory, Vol. CT-16, No. 2, May, 1969.

[5] Kernighan, B. W. and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs." Bell Sys. Tech. J., Vol. 49, pp. 291-307 (1970).

[6] Russo, R. L. "A Heuristic Procedure for the Partitioning and Mapping of Computer Logic Gates," IEEE Trans. Comput., Vol. C-20, No. 12, pp. 1455-1462 (1971).

[7] Schweikert, D. G. and B. W. Kernighan, "A Proper Model for the Partitioning of Electrical Circuits," Proc. of 9th DA Workshop, pp. 57-62 (1972).

[8] Charney, H. R. and D. L. Plato, "Efficient Partitioning of Components," Proc. of 5th DA Workshop, pp. 1-21 (1968).

[9] Donath, W. E. and A. J. Hoffman, "Lower Bounds for Partitioning of Graphs," IBM Journal of Res. and Dev. 17, pp. 420-425 (1973).

[10] Payne, T. S. and W. M. vanCleemput, "Automated Partitioning of Hierarchical Specified Digital System, Proc.19th Design Automation Conf., pp. 183-192 (1982).

[11] McFarland, M. C., "Computer-Aided Partitioning of Behavioral Hardware Descriptions, Proc. 20th Design Automation Conf., pp. 472-478 (1983).

[12] Soukup, J., "Circuit Layout," Proc. of IEEE, Vol. 69, No. 10, pp. 1281-1304 (1981).

[13] Tsukiyama, S., M. Fukui and I. Shirakawa, "A Heuristic Algorithm for a Pin Assignment Problem of Gate Array LSI's," Proc. of ISCAS 84, pp. 465-469 (1984).

[14] Hanan, M. and J. M. Kurtzberg, "Placement Techniques," Chap. 5 in Design Automation of Digital Systems, 1 (Ed. by Breuer, M. A.), pp. 213-282, Prentice Hall (1972).

[15] Hanan, M., P. K. Wolff, Sr. and B. J. Agule, "Some Experimentl Results on Placement Techniques," Proc. of 13th DA Conference, pp. 214-224 (1973).

[16] Goto, S., "An Efficient Algorithm for the Two-Dimensional Placement Problem in Circuit Layout," IEEE Trans. Circuits & Syst., CAS-28, 1, pp. 12-18 (1981).

[17] Hakimi, S. L., "Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph," Oper. Res., 12 pp. 450-459 (1964).

[18] Johnson, D. B. and T. Mizoguchi, "Selecting the kth Element in $X+Y$ and $X_1+X_2+\cdots+X_m$, SIAM J. Computing, 7, 2, pp.141-143 (1978).

[19] Lin, S. and B. Kernighan, "An Effective Algorithm for Travelling-Salesman Problem," Oper. Res., 11, pp. 498-516 (1973).

[20] Stevens, J. E., "Fast Heuristic Techniques for Placing and Wiring Printed Circuit Boards," Ph. D. Dissertation Comp. Sci., Univ. of Illinois (1972).

[21] Goto, S. and E. S. kuh, "An Approach to the Two-Dimensional Placement Problem in Circuit Layout," IEEE Trans. Circuits Syst., CAS-25, pp. 208-214 1978).

[22] Breuer, M. A., "A class of Min-cut Placement Algorithms," Proc. 14th DA Conference, pp. 284-290 (1977).

[23] Kani, K., H. Kawanishi, and A. Kishimoto, "ROBIN: A Building Block LSI Routing Program," Proc. IEEE ISCAS, pp. 658-660 (1976).

[24] Lauther, U., "A Min-cut Placement Algorithm for General Cell Assemblies Based on a Graph Representation," Proc. 16th DA Conference, pp. 1-10 (1979).

[25] Preas, B. T. and C. W. Gwyn, "Methods for Hierachical Automatic Layout of Custom LSI Circuits Masks," Proc. 15th DA Conf., pp. 206-212 (1978).

[26] Sato, K., T. Nagai, H. Shimoyama, and T. Yahara, "MIRAGE-A Simple-Model Routing Program for the Hierachical Layout Design of IC Masks," Proc.16th DA Conf., pp. 297-304 (1979).

[27] Jung, J., S. Goto and H. Hirayama, "A New Approach to the Two-Dimensional Placement Problem of Wire Congestion in Master-Slice LSI Layout Design, Trans. Inst. of Electronics and Communications Engineers of Japan, Vol. J64-A, No. 1, pp. 55-62 (1981).

ANALYSIS OF PLACEMENT PROCEDURES FOR VLSI STANDARD CELL LAYOUT

Mark R. Hartoog

VLSI Technology, Inc.
San Jose, California.

Abstract

This paper describes a study of placement procedures for VLSI Standard Cell Layout. The procedures studied are Simulated Annealing, Min Cut placement, and a number of improvements to Min Cut placement including a technique called Terminal Propagation which allows Min Cut to include the effect of connections to external cells. The Min Cut procedures are coupled with a Force Directed Pairwise Interchange (FDPI) algorithm for placement improvement. For the same problem these techniques produce a range of solutions with a typical standard deviation 4% for the total wire length and 3% to 4% for the routed area. The spread of results for Simulated Annealing is even larger. This distribution of results for a given algorithm implies that mean results of many placements should be used when comparing algorithms. We find that the Min Cut partitioning with simplied Terminal Propagation is the most efficient placement procedure studied.

Overview

Many automatic techniques have been proposed for the standard cell or poly cell placement problem. There has been relatively little careful analysis of the relative merit of these techniques. In the course of the development of a workstation based standard cell placement and routing system, we implemented a number of placement techniques to evaluate their relative performance and the quality of the resulting layout. The main goals we set for the placement section of this system were:

1. The placement should be as completely automatic as possible.
2. The layout should be as close to the minimum area as possible.
3. The layout should have the minimum possible total wire length.
4. The placement procedure should usable for chips with 4000 to 5000 cells on a workstation type computer.

In this paper we evaluate several placement algorithms with respect to these goals.

The techniques we chose to implement were simulated annealing [6] and several variation of min cut techniques [7], coupled with Force Directed Pairwise Interchange [4] as an improvement technique. We rejected out of hand techniques that require good seed placements to obtain optimum results, because this conflicted with our first goal. We chose the min cut techniques because it was fast and good results had been reported using it [1, 8]. We chose simulated annealing because of results reported for this algorithm [2]. We recognized that simulated annealing is too slow to meet our performance goal, but we implemented it anyway in the hope of providing an alternative placement procedure that could be run on a large mainframe computer and would provide very high quality placements. Our implementation of these techniques is described in the following sections.

Simulated Annealing

Simulated annealing is a Monte Carlo technique for the solution of combinatorial optimization problems [6]. Suppose `c(i)` is some cost function associated with some placement state `i`. In simulated annealing you randomly generate a new state `j` by a perturbation to the current state `i`. If `c(j) <= c(i)`, then the new state `j` is always accepted and replaces the old state `i`. If `c(j) > c(i)`, then the new state `j` may still be accepted if `r < exp( -(c(j)-c(i))/kT )` where `r` is a random number between 0 and 1, `k` is Boltzmann's constant, and `T` is the *temperature* of the system. The temperature of the system is initially set to some large value and progressively reduced through what is usually called a *cooling schedule*. If a large number of states are generated at each temperature and the temperature is very slowly reduced, then simulated annealing should produce solutions with close to optimal values of the cost function.

Simulated annealing has been applied to a wide variety of optimization problems and very good results have been reported for the standard cell placement problem [2]. Our adaptation of simulated annealing is closely modeled after that described by Seehen and Sangiovanni-Vincentelli [2]. We begin with a randomly generated placement. We allow cells to overlap, especially at high temperatures. Our cost function is based on the half perimeter bounding box estimate of the wire length, but includes penalty terms for

0738-100X/86/0000/0314$01.00 ©1986 IEEE

cell overlaps and uneven row lengths. We gradually increased the penalty for overlaps and uneven row lengths as the temperature decreased to encourage overlaps at high temperature, but insure a good legal placement at low temperature. New states are generated both by moving a single cell, changing the orientation of a cell, or interchanging two cells. For most of the tests in this study we generated 100 states for each cell at each temperature step. In some of the larger cases we used 200 states per cell. In all our experiments we generated five moves for each interchange and tried an orientation change whenever a move was rejected. We used an exponential cooling schedule given by `T(1) = 0.95 * T(1-1)` where `T(1)` is the temperature at the next step and `T(1-1)` is the temperature at the previous step.

Min Cut

Min Cut is a placement technique based on graph partitioning methods [7]. You start with a group of cells to be placed into a rectangular area. The area is divided into two equal areas either vertically or horizontally and the objective is to divide the group of cells into two approximately equal size groups such that the number of signals that cross the cut line between the two groups is minimized. You first apply this to all the cells to be placed, then the process is successively repeated on each of the groups, dividing the area they occupy into two areas, first horizontally, then vertically, until each group contains less than a specified number of cells. In the standard cell case, the cells then are formed into rows.

The graph or network partitioning problem is known to be NP-complete [10]. We used the heuristic algorithm proposed by Fiduccia and Mattheyses [3]. This heuristic produces good solutions with a near linear time complexity which makes it very attractive for large placement problems. The Fiduccia and Mattheyses algorithm is an optimization method. You start with a arbitrary partitioning and move cells one at a time to the other group. In a single pass all cells that are free to move are moved to the other set once, including those that increase the cut count. The best result is always remembered and restored at the end of each pass. When two successive passes yield no improvement, the process terminates.

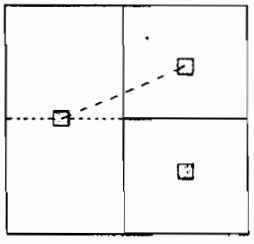


Figure 1

Improvements to Min Cut

In addition to the basic min cut algorithm, we implemented a number of enhancements to it. Because the Fiduccia and Mattheyses algorithm is an optimization technique, it can get stuck in a local minimum. A number of authors have proposed using random perturbations to break out of these local minima. To test whether using random perturbations would yield better or more consistent results, we modified that basic min cut procedure. When two successive passes yield no improvement, we saved the current solution and then randomly selected a percentage of the cells and moved them to the opposite side of the cut line and started the partitioning process again. When two successive passes yielded no improvement again we compared this with our previous result and saved the best result. The process terminated when `N` random perturbation failed to produce a better result than we currently had. Tests where done with values of `N` in the range 1 to 40.

Another enhancement is a method proposed by Dunlop and Kernighan [1] to estimate the total count of all the signals that cross a given cut line including the effect of cells external to the group being partitioned. Dunlop and Kernighan named this *terminal propagation* (TP). At a given stage in the partitioning you position all the cells in the center of the area they will occupy. Then you examine all signals that connect cells in the group being partitioned to external cells. You estimate where routing for this signal will cross the boundary of the group being partitioned. If it crosses the boundary anywhere but near the cut line, the cells on that net are biased towards the side of the cut line the routing intersects by adding a cell to the group which is fixed on that side. Thus even if all the other cells connected to this net are moved to the other side of the cut line, the net is still counts as being cut. Figure 1 shows an example of this. The group of cells was first partitioned horizontally into two groups, then the right group was partitioned vertically. Now the group on the left is being partitioned vertically. The dashed signal net will be counted as being cut unless all the cells connected to it are placed in the top-left area.

For more than 2 point nets, Dunlop and Kernighan [1] computed a rectilinear Steiner tree and found its intersection with the boundary as illustrated in figure 2. In this figure 4 of

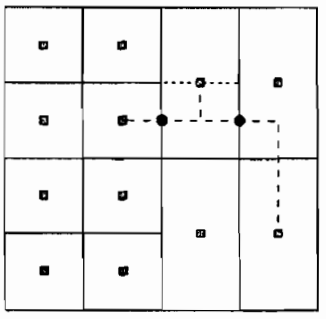


Figure 2

the 8 rectangular areas have already been partitioned vertically. The area with the dashed cut line is currently being partitioned. All the cells have been located at the center of their areas. The dashed 3 point Steiner tree intersects the boundary of the area being partitioned at two points, both below the cut line. This means a cell will be added to this net frozen in the group below the cut line.

We have implemented this procedure and call it *Steiner Tree TP*. The problem of computing minimum length rectilinear Steiner Trees is NP-complete and does not have unique solutions either. We have used a fast heuristic procedure which produces low cost solutions [9].

We have also implemented a procedure we call *Simple TP*. In this procedure you introduce a cell for a net on one side of the cut line, if the net has any pins on that side of the cut line. Simple TP and Steiner Tree TP are identical for 2 pin nets, but Simple TP is substantially faster because it avoids all the Steiner Tree calculation for 3 pin or greater nets.

Placement Improvement

Min Cut partitioning is an initial placement procedure. It only generates approximate positions for cells and does not specify orientation at all. We used *Force Directed Pairwise Interchange* (FDPI) [4] as a placement improvement method. In this technique you compute a force vector `F(1)` for the cell `1` which is the sum over all `j` of `C(1,j) * D(1,j)`. In this expression `C(1,j)` is the number of nets connecting cells `1` and `j` and `D(1,j)` is the distance between cells `1` and `j`. The force vector of a cell **A** selected for interchange is used to compute a target location where the total force on the cell would be zero. A small region around that target is searched for the cell **B** whose target location is closest to the current location of cell **A**. Cells **A** and **B** are then interchanged. If the wire length is reduced, the interchange is accepted, otherwise the cells are restored to their original location. In each pass of placement improvement every cell is interchanged once. Also, each cell is flipped to the other permitted orientation once each pass to see if this reduces the wire length. The process terminates when a complete pass fails to reduce the total wire length by more than a fixed percentage, usually 1 percent for this study. Hanan, et al. [5] showed that FDPI is an efficient algorithms for wire length directed placement improvement.

Analysis Method

To investigate these placement algorithms we placed and routed a number of test circuits using the various placement algorithms. For all of our tests we used a double layer metal cell library. Cells were formed into horizontal rows with routing channels between them. All cells were 72 microns high and all pins were available on both the top and bottom of the cell. The horizontal routing layer was metal and the vertical routing layer was metal2. The cells were pre-analysed to find locations where metal2 could be routed vertically over them. Also the pins on the top and bottom of a cell were freely used to provide vertical routing. The result

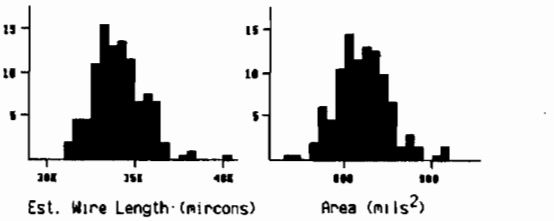


Figure 3

was that additional feed through cells were almost never required to provide additional vertical routing capacity.

In this study we used the half perimeter bounding box wire length after placement and the total area as criterion for evaluating the results. The actual routed wire length was not used because of the use of existing routing inside the cells for much of the vertical routing between rows. This caused the actual routed wire length to occasionally come out less than the estimated wire length. The total area is, of course, not just functions of the placement, but also depend on the routing algorithms. We use a global router to make coarse routing decision such as whether to use the pin on the top or bottom of a cell and assign feed throughs to nets. The global router attempts to route nets so as to minimize channel density. After global routing a channel router is used to perform the actual track assignment and generation of wires. The channel router is based on the YACR algorithm [11] and routes most channels at the theoretical minimum density.

Both the Min Cut algorithm and FDRI are optimization techniques whose final result are a function of the initial state and or the order information appears in the data structures. In the case of min cut partitioning, the initial arbitrary partitioning determines the final result. A different initial partitioning may sometimes yield a better or worse final placement.

Figure 3 shows a histogram of results of 200 placements of a small circuit that contains 112 cells and 121 nets using Min Cut intial placement with Steiner Tree TP and FDPI placement improvement. The input netlist was randomly reordered for each of these placement and routings so that a different arbitrary intial partitioning was used to start the min cut optimization. This figure shows an approximately

Table 1

Circuit	cells	nets	pins per net	number of tests	Wire Length (microns) Mean	Wire Length (microns) Sigma	Area (mils**2) Mean	Area (mils**2) Sigma
A	112	121	2 95	200	33,847	1439	816	31
B	144	176	3 44	20	64,928	2477	1708	55
C	576	714	3 15	50	314,275	13,538	7152	241
D	873	929	3 19	5	685,192	24,772	14,544	544

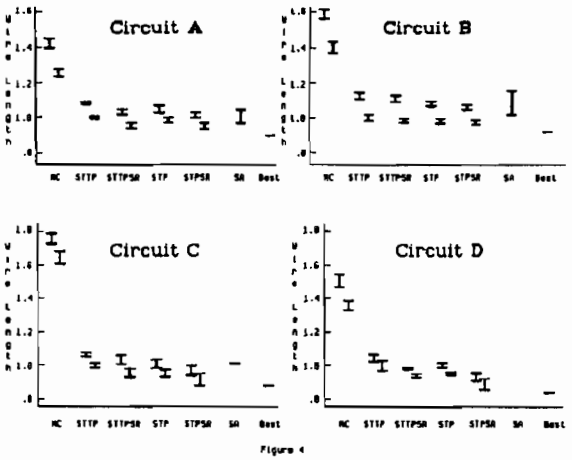


Figure 4

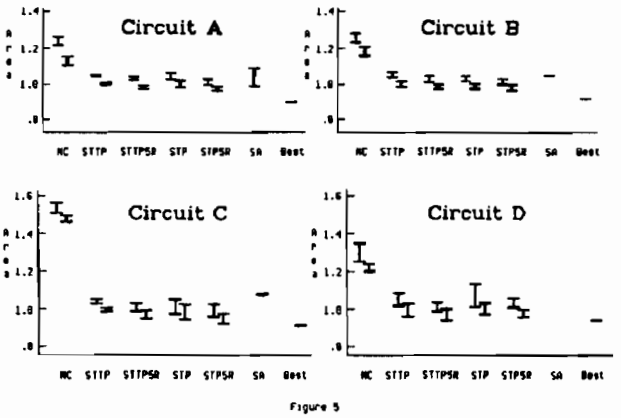


Figure 5

Gaussian distribution for both the half perimeter bounding box estimated wire length and the area of the layout after routing. The mean and standard deviation (sigma) of these results are listed in table 1 for this case (circuit A) and 3 other test circuits. Circuits C and D are complete standard Cell IC's, while circuits A and B are cells used in a larger design. It is clear from table 1 that the results obtained for circuit A are typical of the results for other circuits.

The standard deviation of the estimated wire length is typically 4% of the total wire length and the standard deviation of the area is between 3% and 4% of the total area. This implies that considerable caution must be used in evaluating the results of heuristic optimization algorithms such as these. Even differences as large as 10% between a single placement and routing of a circuit with different algorithms may not be significant.

Since simulated annealing is a Monte Carlo optimization technique, you would expect it to produce a range of results also. Using more temperature steps in the cooling curve and more random states at each temperature step should improve the consistency. Because simulated annealing is slow we have not been able to run large numbers of tests using that algorithm. The results of 6 placements of Circuit A using a cooling function of `T(i) = 0.95 * T(i-1)` and 100 states per cell at each temperature step gave standard deviation of 4.8% and 5.9% for the estimated wire length and area respectively. Clearly the results of a single simulated annealing placement must also be interpreted with caution.

In evaluating the results of different algorithms, we have used the mean and standard deviation of the estimated wire length and area. Clearly we are interested in finding the algorithm that typically produces the best layout, but we also value consistency, so algorithms that have smaller standard deviations are also of interest. Typically we placed and routed circuits A and B 20 times with each algorithm, which gives a good determination of the mean result and an estimate of the standard deviation. Because of the time required, we typically placed and routed the larger circuits C and D only 5 times, which gives a reasonable determination of the mean result, but only a poor estimate of the standard deviation. Because simulated annealing was so slow, only a small number of tests was done with that algorithm.

Results

The main results are displayed in figures 4 and 5. The algorithms shown in this figure are Min Cut (MC), Min Cut with Steiner Tree TP (STTP), Min Cut with Steiner Tree TP and 5 random perturbations, (STTP5R), MIN Cut with Simple TP (STP), Min Cut with Simple TP and 5 random perturbations (STP5R) and Simulated Annealing (SA), and the best result of any single placement by any algorithm (Best). For the algorithms other than Simulated Annealing the results are shown both before and after FDPI placement improvement. The result after FDPI placement improvement is always lower and slightly to the right. The bars in these figures represent the 95% confidence limits on the determination of the mean result for that algorithm. This means that 95% of the time the mean of an infinite number of placement and routing tests would lie within that error bar. The size of the error bars is mostly determined by the number of tests used to find the mean value. The results were normalized relative to the result of the STTP algorithm after FDPI, because this was the best determined point. Figures 4 and 5 clearly confirms the conclusion of Dunlop and Kernighan [1] that Terminal Propagation is a significant improvement in the Min Cut placement algorithm. It is also evident from these figures that Simple Terminal Propagation is just as good, if not better, than Steiner Tree Terminal Propagation. This result is somewhat surprising, since one would expect the Steiner Tree model be a more accurate representation of actual routing. In any event Steiner Tree computations appear to be unnecessary, and this improves the performance of our implementation of this algorithm by about 10%.

Random perturbations appear to make only a small improvement in the average results of the Min Cut algorithm at a considerable performance penalty. Figure 6 shows the effect on the bounding box wire length of increasing the number of random perturbations used in the Min Cut algorithm from 0 to 40 for circuits A and C. The area shows essentially the same results. When the ultimate quality

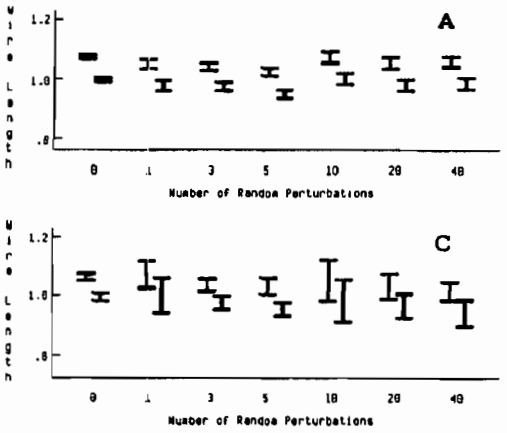


Figure 6

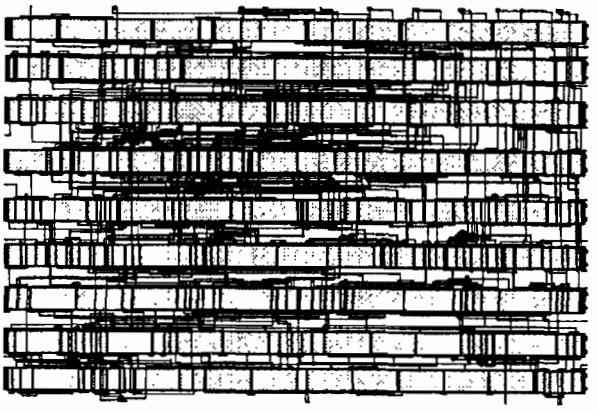


Figure 8

results are desired, then using the Min Cut with random perturbations may yield better results, but simply placing and routing the circuit a number of times and taking the best result could be better than one run with random perturbations.

We had expected Simulated Annealing to produce better results than any of the variations of the Min Cut algorithm. Surprisingly, it did not. There are several arbitrary parameters used in Simulated Annealing. These include the relative scale of overlap penalties to wire length in the cost function and some of the details of how new states are generated. We tried extensive experiments to find optimal values, but no claim is made here that in fact the "best" values were chosen. Finding optimal values of these parameters is very time consuming in light of the typical 5% standard deviation in results from a single run. Circuits should actually be placed and routed 10 to 20 times with each value of a parameter before conclusions are drawn. Clearly this is only possible with small circuits or very fast computers. Our implementation of Simulated Annealing is at least 100 times slower than our Min Cut placement method and does not produce any better layouts. While we believe Simulated Annealing will continue to be of considerable theoretical interest, it does not appear to be useful in a production environment, even when the highest quality results are desired.

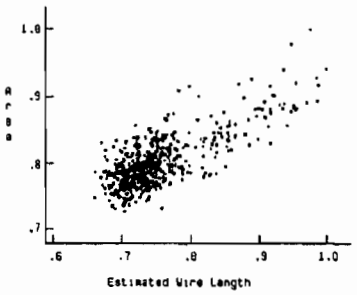


Figure 7

In this study we used both estimated bounding box half perimeter wire length and area after routing as evaluation criterion for our placement algorithms. What is the relationship between these two quantities. Figures 7 shows a plot of the total area as a function of the estimated wire length for 547 placement and routings of circuit A. Clearly there is a correlation between final routed area and the estimated wire length, but it is far from a perfect correlation. In our experience most users of our placement and routing system would prefer the minimum area, with reasonably short wire lengths to the minimum wire length with reasonably small area. This indicates we should move away from heavy dependence on estimated wire length as a placement cost function and move towards cost functions more closely correlated with routing area. This was in fact one of the main arguments for the Min Cut placement algorithm [7]. The FDPI placement improvement, however, only addresses wire length.

Conclusions and Future Work

We have found that the Min Cut partitioning placement method coupled with a simplied version of Terminal Propagation provides the best results as an initial placement method. When combined with a fast placement improvement method, like Force Directed Pairwise Interchange, one has a very efficient placement procedure for standard cells. Figure 8 shows an example of standard cell layout produced using these procedures.

Figure 9 show the CPU time for placement and placement improvement of a number of circuits on an Apollo DN660 with 4 megabytes of main memory (this should be similar to a VAX 780 in performance) as function of the number of cells in the circuit. All the circuits had slightly more nets than cells. The elapsed time was always slightly longer than the CPU time. An analysis of this data indicates that the CPU time grows as the number of cells to 1.6 power. Clearly these algorithms meet the goal of being able to handle 4000 to 5000 cell circuits on a workstation type computer and we should be able to extend these algorithms to 10,000 to 20,000 cell circuits using mainframe computers or the next generation of workstation computers.

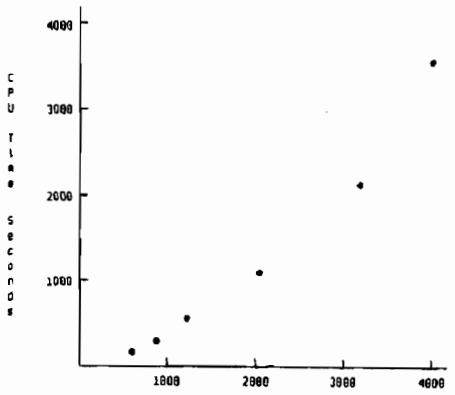


Figure 9

When the best layout possible is desired, then random perturbation in the solution of the Min Cut Partitioning problem produces slightly better results, although at a substantial performance penalty. All of the heuristic optimization techniques investigated in this study yield distributions of wire length and area with standard deviations of 3 to 4 percent. It is not clear whether it is more computationally efficient to place and route a circuit 5 to 10 times and take the best result or whether to use random perturbations in the Min Cut partitioning.

In this study, time prevented us from considering Force Directed and related initial placement techniques [12, 13]. Very good results have also been reported for these methods and it seems possible that they produce more consistent results than the optimization techniques investigated in this paper. Force Directed Pairwise Interchange is a fast, simple placement improvement method. It only addresses wire length and makes no attempt to address routing congestion. We have been able to achieve good result with it because the our initial placements are quite good. We expect that smaller areas could be achieved with a placement improvement method that addressed both wire length and routing congestion. We hope to be able to report on these topics at a future time.

References

1. A. E. Dunlop and B. W. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits", *IEEE Trans on CAD*, vol CAD-4, January 1985.

2. C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package", *Proc. 1984 Custom Integrated Circuits Conference*, pp. 522-527, 1984.

3. C. M. Fiduccia and R. M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions", *Proc. 19th Design Automation Conference*, 1982.

4. M Hanan and J. M. Kurtzberg, "Placement Techniques", Chap. 5 in *Design Automation of Digital Systems: Theory and Techniques*, Vol. 1, Ed. M A. Breuer, Prentice-Hall, N.J. (1972), pp. 213-282.

5. M Hanan, P. K. Wolff, and B J Agule, "Some Experimental Results on Placement Techniques", *Proc. 13th Design Automation Conference*, pp. 214-224, 1973

6. S. Kirkpatrick, C. D. Gelatt, Jr. and M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, Vol. 220, pp. 671-680, 1983.

7. M. A. Breuer, "Min-Cut Placement", *Jou. Design Automation Fault-Tolerant Computing*, Vol 1, pp. 343-362, 1977.

8. H. Shiraishi and F. Hirose, "Efficient Placement and Routing Techniques for Master Slice LSI", *Proc 17th Design Automation Conference*, pp. 191-197, 1980.

9. F. K. Hwang, "The Rectilinear Steiner Problem", *Jou. Design Automation Fault-Tolerant Computing*, Vol 2, pp. 303-310, 1978.

10. M. R. Garey, D. S. Johnson, and L. Sockmeyer, "Some simplified NP-complete graph problems", *Theor. Computer Sci.*, Vol 1, pp. 237-267, 1976.

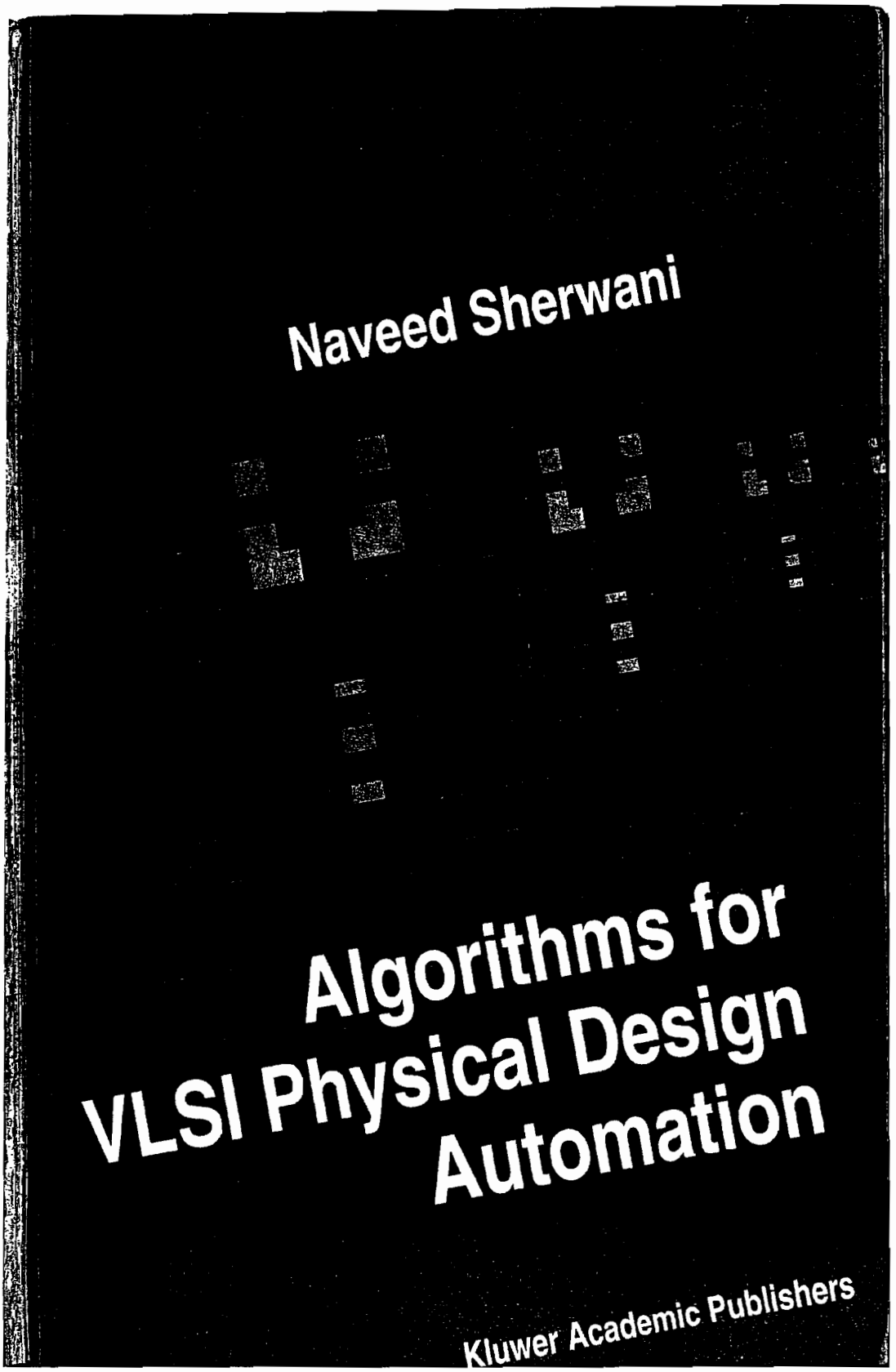
11. A. Sangiovanni-Vincentelli, M. Santomauro, and Jim Reed, "A New Gridless Channel Router: Yet Another Channel Router the Second (YACR-II)", *Proc. International Conference on Computer-Aided Design 84*, pp. 72-75, 1984.

12. J. Blanks, "Near-Optimal Placement Using a Quadratic Objective Function", *Proc. 22nd Design Automation Conference*, pp. 609-615, 1985.

13. C-K. Cheng and E. S. Kuh, "Module Placement Based on Resistive Network Optimization", *IEEE Trans. on CAD*, vol. CAD-3, pp. 218-225, July 1984.

TAB 47


Sherwani
Algorithms for VLSI Physical Design Automation


Naveed Sherwani
Algorithms for VLSI Physical Design Automation
Kluwer Academic Publishers

[redacted]

[redacted]

ALGORITHMS FOR VLSI PHYSICAL DESIGN AUTOMATION

[redacted]

ALGORITHMS FOR VLSI PHYSICAL DESIGN AUTOMATION

by

Naveed A. Sherwani
Western Michigan University

Kluwer Academic Publishers
Boston/Dordrecht/London

Distributors for North America:
Kluwer Academic Publishers
101 Philip Drive
Assinippi Park
Norwell, Massachusetts 02061 USA

Distributors for all other countries:
Kluwer Academic Publishers Group
Distribution Centre
Post Office Box 322
3300 AH Dordrecht, THE NETHERLANDS

Library of Congress Cataloging-in-Publication Data

Sherwani, N. A. (Naveed A.)
Algorithms for VLSI physical design automation / Naveed A. Sherwani.
p. cm.
Includes bibliographical references (p.) and indexes.
ISBN 0-7923-9294-9 (acid-free paper)
1. Integrated circuits--Very large scale integration--Design and construction--Data processing. 2. Computer-aided design. 3. Algorithms. I. Title.
TK874.S455 1993
621.39'5--dc20 92-38496
CIP

Printed on acid-free paper.

Printed in the United States of America

Chapter 5

Placement, Floorplanning and Pin Assignment

After the circuit partitioning phase, the area occupied by each block (subcircuit) can be calculated and the number of terminals (pins) required by each block is known. In addition, the netlist specifying the connections between the blocks is also available. In order to complete the layout, we need to arrange the blocks on the layout surface and interconnect their pins according to the netlist. The arrangement of blocks is done in the placement phase, while interconnection is completed in the routing phase. In the placement phase, blocks are assigned a specific shape and are positioned on a layout surface, in a such a fashion that no two blocks are overlapping and enough space is left on the layout surface to complete the interconnections between the blocks. The blocks are positioned so as to minimize the total area of the layout. In addition, the locations of pins on each block are also determined.

The input to the placement phase is a set of blocks, the number of terminals for each block and the netlist. If the layout of the circuit within a block has been completed then the dimensions of the block are also known. The blocks for which the dimensions are known are called *fixed* blocks and the blocks for which dimensions are yet to be determined are called *flexible* blocks. Thus during the placement phase, we need to determine an appropriate shape for each block (if shape is not known), location of each block on the layout surface, and determine the locations of pins on the boundary of the blocks. The problem of assigning locations to the fixed blocks on a layout surface is called the *Placement* problem. If some or all of the blocks are flexible then the problem is called the *Floorplanning* problem. Hence, the placement problem is a restricted version of the floorplanning problem. The terminology is slightly confusing as floorplanning problems are placement problems as well but these terminologies have been widely used and accepted. It is desirable that the pin locations are identified at the same time when the block locations are fixed. However, due to the complexity of the placement problem, the problem of identifying the pin locations for the blocks is solved after the locations of all

the blocks are known. This process of identifying pin locations is called *pin assignment.*

Placement phase is very crucial in overall physical design cycle. It is due to the fact, that an ill-placed layout cannot be improved by high quality routing. In other words, the overall quality of the layout, in terms of area and performance is mainly determined in the placement phase.

There are several factors that are considered by the placement, floorplanning and pin assignment algorithms. These factors are discussed below:

1. **Shape of the blocks:** In order to simplify the problem, the blocks are assumed to be rectangular. The shapes resulting from the floorplanning algorithms are mostly rectangular for the same reason. The floorplanning algorithms use *aspect ratios* for determining the shape of a block. The aspect ratio of a block is the ratio between its height and its width. Usually there is an upper and a lower bound on the aspect ratios, restricting the dimensions that the block can have. More recently, other shapes such as L-shapes have been considered, however dealing with such shapes is computationally intensive.

2. **Routing considerations:** One of the objectives of the placement and floorplanning algorithms is to estimate the area required for routing. The blocks are placed in a manner such that there is sufficient routing area between the blocks, so that routing algorithms can complete the task of routing of nets between the blocks. If complete routing is not possible, placement phase has to be repeated.

3. **Placement for high performance circuits:** For high performance circuits the blocks are to be placed such that all critical nets can be routed within their timing budgets. In other, the length of critical paths must be minimized. The placement process for high performance circuits is also called as *performance driven* placement.

4. **Packaging considerations:** All of these blocks generate heat when the circuit is operational. The heat dissipated should be uniform over the entire surface of the group of blocks placed by the placement algorithms. Hence, the placement algorithms must distribute the blocks which generate a large amount of heat while placing them. This might conflict with the objective for high performance circuits and some trade off has to be made.

5. **Pre-placed blocks:** In some cases, the locations of some of the blocks may be fixed, or a region may be specified for their placement. For example, in high performance chips, the clock buffer may have to be located in the center of the chip. This is done with the intention to reduce the time difference between arrival time of the clock signal at different blocks. In some cases, a designer may specify a region for a block, within which the block must be placed.

In this chapter, we will discuss placement, floorplanning and pin assignment problems in different design styles. Section 5.1 discusses the placement problem and the proposed placement algorithms. Section 5.2 presents algorithms for the floorplanning problems, while pin assignment is discussed in section 5.3. In section 5.4, we discuss integrated approach to these problems.

5.1 Placement

Placement is a key step in physical design cycle. A poor placement not only takes up larger areas and degrades performance, it also leads to a difficult or sometimes impossible routing task. The placement of block occurs at three different levels.

1. **System level placement:** At system level, the placement problem is to place all the PCBs together so that the area occupied is minimum. At the same time, the heat generated by each of the PCBs should be dissipated properly so that the system does not malfunction due to overheating of some component.

2. **Board level placement:** At board level, all the chips on a board along with other solid state devices have to be placed within a fixed area of the PCB. All blocks are fixed and rectangular in shape. In addition, some blocks may be pre-placed. The PCB technology allows mounting of components on both sides. There is essentially no restriction on the number of routing layers in PCB. Therefore, generally speaking, no matter how badly the components are placed the nets can always be routed. The objective of board-level placement algorithms is to minimize the number of routing layers and satisfy system performance requirements. For high performance circuits, the critical nets should have lengths which are less than a specified value and hence the placement algorithms should place the critical components closer together. Another key placement problem is the temperature profile of the board. The heat dissipation on a PCB should be uniform, i.e., the chips which generate maximum heat should not be placed closer to each other. If MCMs are used instead of PCBs, then the heat dissipation problem is even more critical, since chips are placed closer together on a MCM.

3. **Chip level placement:** At chip level, the problem can be either placement or floorplanning along with pin assignment. The blocks are either flexible or fixed, and some of them may be pre-placed. The key difference between the board level placement problem and the chip level placement is the limited number of layers that can be used for routing in a chip. In addition, the circuit is fabricated only on one side of the substrate. This implies that some 'bad' placements maybe unroutable. However, the fact that a given placement is unroutable will not be discovered until routing is attempted. This leads to very costly delays in completion of the design. Therefore, it is very important to accurately determine the routing areas

in the chip-level placement problems. Generally, two or three layers are used for routing, however, chips with three layer routings are more expensive to fabricate. The objective of a chip-level placement or floorplanning algorithm is find a minimum area routable placement of the blocks. In some cases, a mixture of macro blocks and standard cells may have to be placed together. These problems are referred to as ***Mixed block and cell*** placement and floorplanning problems. At chip level, if the design is hierarchical then the placement and floorplanning is also carried out in a hierarchical manner. The hierarchical approach can greatly simplify the overall placement process.

In the following sections, we will discuss the chip-level placement. The placement problem for MCMs, which is essentially a performance driven Board level placement problem, will be discussed in Chapter 12.

5.1.1 Problem Formulation

The placement problem can be stated as follows: Given an electrical circuit consisting of fixed blocks, and a netlist interconnecting terminals on the periphery of these blocks and on the periphery of the circuit itself, construct a layout indicating the positions of each block such that all the nets can be routed and the total layout area is minimized. The objective for high performance systems is to minimize the total delay of the system, by minimizing the lengths of the critical paths. It is usually approximated by minimization of the length of the longest net. This problem is known as the *performance (timing) driven placement problem.* The associated algorithms are called *performance (timing) driven placement algorithms.*

The quality of a placement is based on several factors:

1. layout area.
2. completion of routing,
3. circuit performance, and

The layout area and the routability of the layout are usually approximated by the topological congestion, known as ***rat's nest***, of interconnecting wires. Consider the simple example in Figure 5.1. Two different placements for this example is shown in Figure 5.2. The topological congestion in Figure 5.2(a) is much less than that of in Figure 5.2(b). Thus, the placement given in Figure 5.2(a) can be considered more easily routable than the placement given in Figure 5.2(b). In many cases, several objectives may contradict each other. For example, minimizing layout area may lead to increased maximum wire length and vice versa.

Let us formally state the placement problem. Let $B_1, B_2, \ldots, B_n$ be the blocks to be placed on the chip. Each $B_i, 1 \leq i \leq n$, has associated with it a height h_i and a width w_i. Let $\mathcal{N} = \{N_1, N_2, N_3, \ldots, N_m\}$ be the set of nets representing the interconnection between different blocks. Let $\mathcal{Q} =$

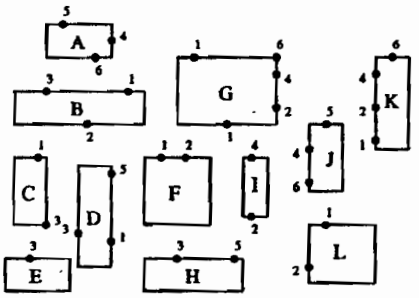


Figure 5.1: Blocks and set of interconnecting nets.

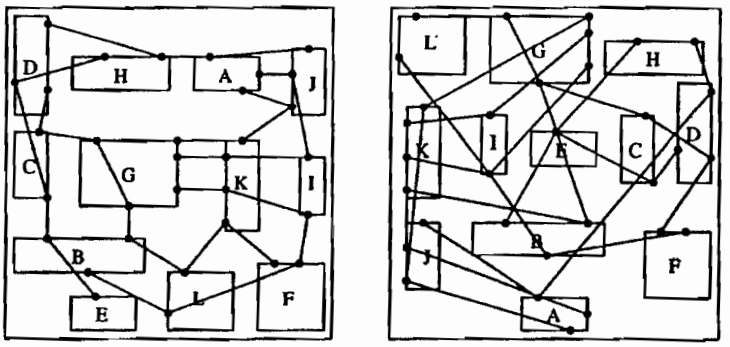


Figure 5.2: Two different placements of the same problem.

$\{Q_1, Q_2, \ldots, Q_k\}$ represent rectangular empty areas allocated for routing between blocks. Let L_i denote the estimated length of net N_i, $1 \leq i \leq m$. The placement problem is to find an iso-oriented rectangles for each of these blocks on the plane denoted by $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ such that

1. Each block can be placed in its corresponding rectangle, that is, R_i has width w_i and height h_i,
2. No two rectangles overlap, that is, $R_i \cap R_j = \phi$, $1 \leq i, j \leq n$,
3. Placement is routable, that is, $Q_j, 1 \leq j \leq k$, is sufficient to route all the nets.
4. The total area of the rectangle bounding $\mathcal{R}$ and Q is minimized.

5. The total wirelength is minimized, that is, $\sum_{i=1}^{m} L_i$ is minimized. In the case of high performance circuits, the length of longest net $\max\{L_i \mid i = 1,\ldots,m\}$ is minimized.

The general placement problem is NP-complete and hence, the algorithms used are generally heuristic in nature.

Although the actual wiring paths are not known at the time of placement, however, a placement algorithm needs to model the topology of the interconnection nets. An interconnection graph structure which interconnects each net is used for this purpose. The interconnection structure for two terminal trees is simply an edge between the two vertices corresponding to the terminals. In order to model a net with more than two terminals, rectilinear steiner trees are used as shown in Figure 5.3(a) to estimate optimal wiring paths for a net. This method is usually not used by routers, because of the NP-completeness of steiner tree problem. As a result, minimum spanning tree representations are the most commonly used structures to connect a net in the placement phase. Minimum spanning tree connections (shown in Figure 5.3(b)) allow branching only at the pin locations. Hence, the pins are connected in the form of minimum spanning tree of a graph. *Complete graph* interconnection is shown in (Figure 5.3(c)). It is easy to implement such structures. However, this method causes many redundant interconnections, and results in longer wire length.

The large number of objective functions can be classified into two categories, net metrics and congestion metric. The net metrics deal with the assumption that all the nets can be routed without interfering with other nets or with the components. Usually the length of a net is important as the interconnection delays depend on the length of the wire. The net metrics only quantify the amount of wiring and do not account for the actual location of these wires. The examples of this kind of objective functions are the total length of all nets and the length of the longest net. The congestion metric is used to avoid the buildup of many nets in a particular area leading to congestion. Example of congestion metric is the number of nets that intersects with a routing channel.

The layout surface on which the circuit is to be placed is modeled into either geometric or topological models. For the geometric model, the placement algorithms tend to accept the layout area as a fixed constraint and tend to optimize the interconnections. The geometric models are appropriate for design styles where placement aspects such as size, shape and public pin positions do not change during the layout process such as PCB design. On the other hand, the placement systems which model the layout surface as a topological model assume the constraint to be the completion of interconnections and optimize the layout area. Topological models are appropriate for more flexible design styles such as full custom designs.

5.1.1.1 Design Style Specific Placement Problems

Different design styles impose different restrictions on the layout and have different objectives in placement problems.

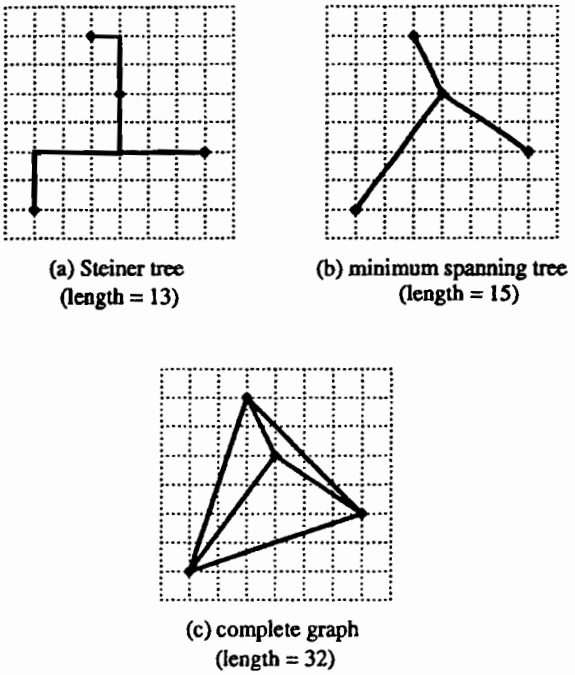


Figure 5.3: Interconnection topologies.

1. **Full custom:** In full custom design style, the placement problem is the packing problem concerned with placing a number of blocks of different sizes and shapes tightly within a rectangular area. There is no restriction on how the blocks can be placed within the rectangle except that no two blocks may overlap. The primary objective is to minimize the total layout area. The irregularity of the block shapes is usually the main cause of unused areas. Since unused area increases the total area, the blocks must be placed so as to minimize unused areas. The objective of minimizing the layout area sometimes conflicts with the objective of minimizing the maximum length of a net. Therefore, in high performance circuit design, additional constraints on net lengths must also be considered.

2. **Standard cells:** The standard cell placement problem is somewhat simpler than the full custom placement problem, as all the cells have the same height. Cells are placed in rows and minimizing layout area is equivalent to minimizing the summation of channel heights and minimizing the width of the widest row. In order to reduce overall area, all rows should have equal widths. The total area consists of area required for the cell rows and the area required for routing or the channel area.

The routing area estimates, which determine the channel height, play a key role in determining the overall area of the design. With the advent of over-the-cell routing, in which the empty spaces over the standard cell rows are used for routing, the channels in standard cells have almost disappeared giving rise to *channelless* standard cell designs. Standard cells are designed so the power and ground nets run horizontally through the top and bottom of cells.

3. **Gate arrays:** As mentioned in the previous chapter, in case of gate arrays, the partitioning of a circuit maps the circuit onto the gates of the gate array. Hence the problem of partitioning and placement is essentially the same in this design style. If partitioning does not actually assign gate locations, then a placement algorithm has to be used to assign subcircuits or gates to the slots on the gate array. Given a set of blocks $B_1, B_2, \ldots, B_n$, and set of slots $S_1, S_2, \ldots, S_r$, $r \geq n$, assign each block B_i to a slot S_j such that no two blocks are assigned to the same slot and the placement is routable. For high performance designs, additional constraints on net lengths have to be added.

   Another situation, where gate array partitioning and placement may be different, is when each 'gate' in the gate array is a complex circuit. In this case, the circuit is partitioned such that each subcircuit is equivalent to a 'gate'. The placement algorithm is then used to find the actual assignment. This happens to be the case in FPGAs and is discussed in Chapter 11.

5.1.2 Classification of Placement Algorithms

The placement algorithms can be classified on the basis of :

1. the input to the algorithms,
2. the nature of output generated by the algorithms, and
3. the process used by the algorithms.

Depending on the input, the placement algorithms can be classified into two major groups: *constructive placement* and *iterative improvement* methods. The input to the constructive placement algorithms consists of a set of blocks of along with the netlist. The algorithm finds the locations of blocks. On the other hand iterative improvement algorithms start with an initial placement. These algorithms modify the initial placement in search of a better placement. These algorithms are typically used in an iterative manner until no improvement is possible.

The nature of output produced by an algorithm is another way of classifying the placement algorithms. Some algorithms generate the same solution when presented with the same problem, i.e., the solution produced is repeatable. These algorithms are called *deterministic* placement algorithms. Algorithms

that function on the basis of fixed connectivity rules (or formulae) or determine the placement by solving simultaneous equations are deterministic and always produce the same result for a particular placement problem. Some algorithms, on the other hand, work by randomly examining configurations and may produce a different result each time they are presented with the same problem. Such algorithms are called as *probabilistic* placement algorithms.

The classification based on the process used by the placement algorithms is perhaps the best way of classifying these algorithms. There are two important class of algorithms under this classification: simulation based algorithms and partitioning based algorithms. Simulation based algorithms simulate some natural phenomenon while partitioning based algorithms use partitioning for generating the placement. The algorithms which use clustering and other approaches are classified under 'other' placement algorithms.

5.1.3 Simulation Based Placement Algorithms

There are many problems in the natural world which resemble placement and packaging problems. Molecules and atoms arrange themselves in crystals, such that these crystals have minimum size and no residual strain. Herds of animals move around, until each herd has enough space and it can maintain its predator-prey relationships with other animals in other herds. The simulation based placement algorithms simulate some of such natural processes or phenomenon. There are three major algorithms in this class: simulated annealing, simulated evolution and force directed placement. The simulated annealing algorithm simulates the annealing process which is used to temper metals. Simulated evolution simulates the biological process of evolution while the force directed placement simulate, a system of bodies attached by springs. These algorithms are described in the following subsections.

5.1.3.1 Simulated Annealing

Simulated annealing is one of the most well developed placement methods available [8, 123, 125, 130, 165, 200, 247, 273, 274, 291, 292]. The simulated annealing technique has been successfully used in many phases of VLSI physical design, e.g., circuit partitioning. The detailed description of the application of simulated annealing method to partitioning may be found in Chapter 4. Simulated annealing is used in placement as an iterative improvement algorithm. Given a placement configuration, a change to that configuration is made by moving a component or interchanging locations of two components. In case of the simple pairwise interchange algorithm, it is possible that a configuration achieved has a cost higher than that of the optimum but no interchange can cause a further cost reduction. In such a situation the algorithm is trapped at a local optimum and cannot proceed further. Actually this happens quite often when this algorithm is used on real life examples. Simulated annealing avoids getting stuck at a local optimum by occasionally accepting moves that result in a cost increase.

```
Algorithm SIMULATED-ANNEALING
begin
    temp = INIT-TEMP;
    place = INIT-PLACEMENT;
    while (temp > FINAL-TEMP) do
        while (inner_loop_criterion = FALSE) do
            new_place = PERTURB(place);
            ΔC = COST(new_place) - COST(place);
            if (ΔC < 0) then
                place = new_place;
            else if (RANDOM(0,1) > e^(-ΔC/T)) then
                place = new_place;
        temp = SCHEDULE(temp);
end.
```

Figure 5.4: The Simulated Annealing algorithm.

In simulated annealing, all moves that result in a decrease in cost are accepted. Moves that result in an increase in cost are accepted with a probability that decreases over the iterations. The analogy to the actual annealing process is heightened with the use of a parameter called *temperature* T. This parameter controls the probability of accepting moves which result in an increased cost. More of such moves are accepted at higher values of temperature than at lower values. The acceptance probability can be given by $e^{\frac{-\Delta C}{T}}$, where ΔC is the increase in cost. The algorithm starts with a very high value of temperature which gradually decreases so that moves that increase cost have lower probability of being accepted. Finally, the temperature reduces to a very low value which causes only moves that reduce cost to be accepted. In this way, the algorithm converges to a optimal or near optimal configuration.

In each stage, the configuration is shuffled randomly to get a new configuration. This random shuffling could be achieved by displacing a block to a random location, an interchange of two blocks, or any other move which can change the wire length. After the shuffle, the change in cost is evaluated. If there is a decrease in cost, the configuration is accepted, otherwise, the new configuration is accepted with a probability that depends on the temperature. The temperature is then lowered using some function which, for example, could be exponential in nature. The process is stopped when the temperature has dropped to a certain level. The outline of the simulated annealing algorithm is shown in Figure 5.4.

The parameters and functions used in a simulated annealing algorithm determine the quality of the placement produced. These parameters and functions include the cooling schedule consisting of initial temperature (*init_temp*), final temperature (*final_temp*) and the function used for changing the temperature (SCHEDULE), *inner_loop_criterion* which is the number of trials at

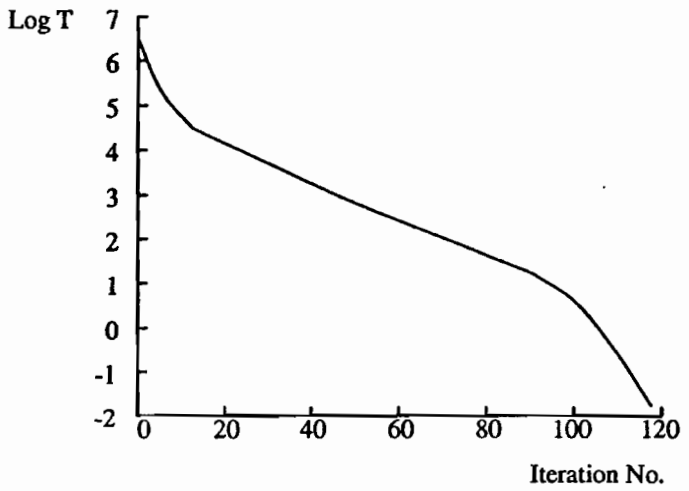


Figure 5.5: Cooling schedule in TimberWolf.

each temperature, the process used for shuffling a configuration (PERTURB), acceptance probability (F), and the cost function (COST). A good choice of these parameters and functions can result in a good placement in a relatively short time.

Sechen and Sangiovanni-Vincentelli developed TimberWolf 3.2, which is a standard cell placement algorithm based on Simulated Annealing [292]. TimberWolf is one of the most successful placement algorithms. In this algorithm, the parameters and functions are taken as follows. For the cooling schedule, $init\_temp = 4000000$, $final\_temp = 0.1$, and SCHEDULE $(T) = \alpha(T) \times T$ where $\alpha(T)$ is a cooling rate depending on the current temperature T. $\alpha(T)$ is taken relatively low when T is high, e.g. $\alpha(T) = 0.8$ when the cooling process just starts, which means the temperature is decremented rapidly. Then, in the medium range of temperature, $\alpha(T)$ is taken 0.95, which means that the temperature changes more slowly. When the temperature is in low range, $\alpha(T)$ is again taken 0.8, the cooling procedure go fast again. In this way, there are a total of 117 temperature steps. The graph for the cooling schedule is shown in Figure 5.5. The value of $inner\_loop\_criterion$ is taken according to the size of the circuit, e.g., 100 moves per cell for a 200-cell circuit and 700 moves per cell for a 3000-cell circuit are recommended in [292]. The new configuration is generated by making a weighted random selection from one of the following:

1. the displacement of a block to a new location,
2. the interchange of locations between two blocks,
3. an orientation change for a block.

The alternative 3 is used only when the new configuration generated by using alternative 1 is rejected. The ratio r of single block displacement to pairwise interchange should be carefully chosen to give a best overall result. An orientation change of a block is simply a mirror image of that block's x-coordinate. The cost function is taken as:

$$COST = cost1 + cost2 + cost3$$

where $cost1$ is the weighted sum of estimate length of all nets, $cost2$ is the penalty cost for overlapping, and $cost3$ is the penalty cost for uneven length among standard cell rows.

$$cost1 = \sum_{i \in nets} [xspan(i) \times HWeight(i) + yspan(i) \times VWeight(i)]$$
$$cost2 = \sum_{i \neq j \in blocks} [overlap(i,j)]^2$$
$$cost3 = \sum_{i \in rows} |ActRowLength(i) - DesRowLength(i)| \times factor$$

Where, $xspan(i)$ and $yspan(i)$ are the horizontal and vertical spans of the minimum bounding rectangle of net i. Horizontal and vertical weights ($HWeight$ and $VWeight$) are introduced so that each net can have different priority to be optimized, e.g. critical nets can have higher priority, and one direction can be favored over another direction. The quadratic function in $cost2$ is used to penalize more heavily on large overlaps than small ones. Actually overlap is not allowed in the placement. However, it takes large amount of computer time to remove all overlapping. So, it is more efficient to allow overlapping during intermediate placement and use a cost function to penalize the overlapping. $ActRowLength(i)$ and $DesRowLength(i)$ are the actual row length and desired row length for the ith row, respectively. The $factor$ is used so that the minimum penalty for the difference in length of rows is $factor$.

The simulated annealing is one of the most established algorithms for placement problems. It produces good quality placement. However, Simulated Annealing is computationally expensive and can lead to longer run times. Therefore, it is only suitable for small to medium sized circuits.

5.1.3.2 Simulated Evolution

Simulated evolution (genetic algorithm) is analogous to the natural process of mutation of species as they evolve to better adapt to their environment. It has been recently applied to various fields. Readers are referred to the chapter 4 for the description of simulated evolution algorithm used in partitioning.

We use the example of gate array placement problem to explain the simulated evolution algorithm. In gate array placement problem, the layout plane is divided into $S = \{S_1, S_2, \ldots, S_r\}$ slots. The problem of placing cells $B = \{B_1, B_2, \ldots, B_n\}$, where $n \leq r$, is to assign some S_j to each B_i, such that no two cells are assigned to the same slot. The algorithm starts with an initial set of placement configurations, which is called the *population*. This initial placement can be generated randomly. The individuals in this population represent a feasible placement to the optimization problem and are actually represented by a string of symbols. The symbols used in the solution string are called

genes. A solution string made up of genes is called a *chromosome.* A *schema* is a set of genes that make up a partial solution. Simulated evolution algorithm is iterative, and each iteration is called a *generation.* During each iteration the individuals of the population are evaluated on the basis of certain *fitness* tests which can determine the quality of each placement. Two individuals (corresponding to two possible placement configurations) among the population are selected as *parents* with probabilities based on their fitness. The better fitness an individual has, the higher the probability that it will be chosen. The operators called crossover, mutation and inversion, which are analogous to the counterparts in the evolution process, are then applied on the parents to combine 'genes' from each parent to generate a new individual called the *offspring.* The offsprings are then evaluated and a new generation is then formed by including some of the parents and the offsprings on the basis of their fitness in a manner that the size of population remains the same. As the tendency is to select high fitness individuals to generate offsprings and the weak individuals are deleted, the next generation tends to have individuals that have good fitness. The fitness of the entire population improves over the generations. That means the overall placement quality improves over iterations. At the same time, some 'bad' genes are inherited from previous generation even though the probability of doing so is quite low. In this way, it is assured that the algorithm does not get stuck at some local optimum. This is the basic mechanism of the algorithm which results in a good placement. The three genetic operators that are used for creating offsprings are discussed below.

1. **Crossover :** Crossover generates offsprings by combining schemata of two individuals at a time. This could be achieved by choosing a random cut point and generating the offspring by combining the left segment of one parent with the right segment of the other. However, after doing so, some blocks may be repeated while some other blocks may get deleted. This problem has been dealt with in many different ways. The amount of crossover is controlled by the crossover rate which is defined as the ratio of the number of offspring produced in each generation to the population size. The crossover rate determines the ratio of the number of searches in regions of high average fitness to the number of searches in other regions.

2. **Mutation:** This operator is not directly responsible for producing new offsprings but it causes incremental random changes in the offspring produced by crossover. The most commonly used mutation is pair-wise interchange. This is the process by which new genes which did not exist in the original generation can be generated. The mutation rate is defined as the percentage of the total number of genes in the population, which are mutated in each generation. It should be carefully chosen so that it can introduce more useful genes, and at the same time do not destroy the resemblance of *offsprings* to their *parents.*

3. **Selection:** After the offspring is generated, individuals for the next generation are chosen based on some criteria. There are many such se-

```
Algorithm GENIE
begin
    no_pop = SIZE-POP;
    no_offspring = no_pop × P_ψ;
    (* P_ψ stands for the crossover rate. *)
    pop = CONSTRUCT-POP(no_pop);
    for (i = 1 to no_pop) do
    (* Evaluate score of each individual in
       the population on the basis of its fitness. *)
        SCORE(pop(i));
    for (i = 1 to no_generation) do
    (* Generate no_generation generations *)
        for (j = 1 to no_offspring) do
            (x, y) = CHOOSE-PARENT(pop)
            offspring(j) = GENERATE(x, y);
            SCORE(offspring(j));
        pop = SELECT(pop, offspring, no_pop);
        for (j = 1 to no_pop) do
            MUTATE(pop(j));
    return highest scoring configuration in population;
end.
```

Figure 5.6: The Simulated Evolution algorithm.

lection functions used by various researchers. In *competitive selection* all the parents and offsprings compete with each other and the fittest individuals are selected so that the population remains constant. In *random selection* the individuals for the next generation are randomly selected so that the population remains constant. This could be advantageous considering the fact that by selecting the fittest individuals the population converges to individuals that share the same genes and the search might not converge to a optimum. However, if the individuals are chosen randomly, there is no way to gain improvements from older generation to new generation. By compromising both methods, *stochastic selection* makes selections with probabilities based on the fitness of each individual.

An algorithm developed by Cohoon and Paris [49] is shown in Figure 5.6. The scoring function is chosen to account for total net lengths and to penalize the placement with high wiring density in the routing channels. The score is given by:

$$\sigma = \frac{1}{2}\sum_{i \in nets} length(i) + \sum_{i \in HChannels} h_i'^2 + \sum_{i \in VChannels} v_i'^2$$

where

$$h_i' = \begin{cases} h_i - h_{avg} - h_{sd} & \text{if } h_i > h_{avg} - h_{sd} \\ 0 & \text{otherwise} \end{cases}$$

$$v_i' = \begin{cases} v_i - v_{avg} - v_{sd} & \text{if } v_i > v_{avg} - v_{sd} \\ 0 & \text{otherwise} \end{cases}$$

where, h_i (v_i) is the number of nets intersecting horizontal (vertical) channel i. h_{avg} (v_{avg}) is the mean of h_i (v_i). h_{sd} (v_{sd}) is the standard deviation of h_i (v_i).

The parent choosing function is performed alternatively as either selecting parents with probabilities proportional to their fitness or selecting parents with probabilities proportional to their fitness and an additional constraint such that they have above average fitness. Two crossover operators can be used. One selects a random cell C_s and brings the four closest neighbors in parent 1 into neighboring slots in parent 2. At the same time, the cells in these slots in parent 2 are pushed outward until vacant slots are found. The other one selects a square of $k \times k$ cells from parent 1 where k is a random number with mean of 3 and variance of 1, and copy the square into parent 2. The result of this copying would result in the loss of some cells. So, before copying, the cells in parent 2 that are not part of square are being pushed outward into some vacant slots.

One possible mutation function is to use a greedy technique to improve the placement cost. It selects a cell C_i on a net N_j and searches the cell C_k on the same net that is farthest from cell C_i. C_k is then brought close to the cell C_i. The cell which needs to be removed from that slot is pushed outward until an vacancy is found.

Besides the implementation described above, there are other implementations, e.g. the genetic approach developed by Chan, Mazumder and Shahookar, which uses a two-dimensional bitmap chromosome to handle the placement of macro cells and gate arrays [31]. In addition, the simulated evolution was investigated in [32, 192, 193, 293, 294].

5.1.3.3 Force Directed Placement

Force directed placement explores the similarity between placement problem and classical mechanics problem of a system of bodies attached to springs. In this method, the blocks connected to each other by nets are supposed to exert attractive forces on each other. The magnitude of this force is directly proportional to the distance between the blocks. According to Hooke's law, the force exerted due to stretching of the springs is proportional to the distance between the bodies connected to the spring. If the bodies were allowed to move freely, they would move in the direction of the force until the system achieved equilibrium. The same idea is used for placing the blocks. The final configuration of the placement of blocks is the one in which the system achieves equilibrium.

In [261], Quinn developed a placement algorithm using force directed method. In this algorithm, all the blocks to be placed are considered to be rectangles. These blocks are classified as movable or fixed move. Let $B = \{B_1, B_2, \ldots, B_n\}$ be the blocks to be placed, and (x_i, y_i) be the Cartesian coordinates for B_i. Let $\Delta x_{ij} = |x_i - x_j|$, $\Delta y_{ij} = |y_i - y_j|$. $\Delta d_{ij} = \sqrt{(\Delta x_{ij})^2 + (\Delta y_{ij})^2}$. Let F_x^i (F_y^i) be the total force enacted upon B_i by all the other blocks in the x-direction (y-direction). Then, the force equations can be expressed as:

$$F_x^i = \sum_{j=1}^{n}[-k_{ij} \times \Delta x_{ij}]$$
$$F_y^i = \sum_{j=1}^{n}[-k_{ij} \times \Delta y_{ij}]$$

where $i = 1, 2, \ldots, n$, k_{ij} = attractive constant between blocks B_i and B_j and $k_{ij} = 0$ if $i = j$. The blocks connected by nets tend to move toward each other, and the force between them is directly proportional to the distance between them. On the other hand, the force model does not reflect the relationship between unconnected blocks. In fact, the unconnected blocks tend to repel each other. So, the above model should be modified to include this repulsion effects. Since the formulation of the force equation in x-direction is the same as in y-direction, in the following, only the formulation in x-direction will be discussed.

$$F_x^i = \sum_{j=1}^{n}[-k_{ij} \times \Delta x_{ij} + \delta k_{ij} \times R \times \Delta x_{ij}/\Delta d_{ij}], \quad i = 1, 2, \ldots, n$$

where $\delta k_{ij} = 1$ when $k_{ij} = 0$, and $\delta k_{ij} = 0$ when $k_{ij} = 1$. R is the repulsion constant directly proportional to the maximum of k_{ij} and inversely proportional to n.

In addition, it is also desirable to have the center of all movable blocks be in some predetermined physical location (usually the geometric center of the layout plane) so that the placement of blocks is balanced. Physically, it is equivalent to have the forces acted upon the set of all movable blocks being removed. Suppose there are m movable blocks. Then, the force equations become:

$$F_x^i = \sum_{j=1}^{n}[-k_{ij} \times \Delta x_{ij} + \delta k_{ij} \times R \times \Delta x_{ij}/\Delta d_{ij}] - F_{ext}, \quad i = 1, 2, \ldots, n$$

where F_{ext} is the total external force acted upon the set of all movable blocks by the fixed blocks and $F_{ext} = \{\sum_{i=1}^{m}\sum_{j=m+1}^{n}[-k_{ij} \times \Delta x_{ij} + \delta k_{ij} \times R \times \Delta x_{ij}/\Delta d_{ij}]\}/m$.

The placement problem now becomes a problem in classical mechanics and the variety of methods used in classical mechanics can be applied. To solve for the set of force equations, one of the methods is to set the potential energy equal to $\sum_{i=1}^{n}[F_x^{i2} + F_y^{i2}]$, and apply the unconstrained minimization method, i.e., Fletcher-Reeves method [98], since the solution of the force equations correspond to the state of zero potential energy of the system.

Besides the implementation presented by Quinn [261], there are various implementations [5, 121, 137, 139, 247, 262].

5.1.3.4 Comparison of Simulation Based Algorithms

Both the simulated annealing and simulated evolution are iterative and probabilistic methods. They can both produce optimal or near-optimal placements, and they are both computation intensive. However, the simulated evolution has an advantage over the simulated annealing by using the history of previous trial placements. The simulated annealing can only deal with one placement configuration at a time. In simulated annealing it is possible that a good configuration maybe obtained and then lost when a bad configuration is introduced later. On the other hand, the good configuration has much better chance to survive during each iteration in simulated evolution since there are more than one configurations being kept during each iteration. Any new configuration is generated by using several configurations in simulated evolution. Thus, history of previous placements can be used. However, the genetic method has to use much more storage space than the simulated annealing since it has to memorize all individual configurations in the population. Unlike simulated annealing and simulated evolution, force directed placement is applicable to general designs, such as full custom designs. The force-directed methods are relatively faster compared to the simulated annealing and genetic approaches, and can produce good placement.

5.1.4 Partitioning Based Placement Algorithms

This is an important class of algorithms in which the given circuit is repeatedly partitioned into two subcircuits. At the same time, at each level of partitioning, the available layout area is partitioned into horizontal and vertical subsections alternately. Each of the subcircuits so partitioned is assigned to a subsection. This process is carried out till each subcircuit consists of a single gate and has a unique place on the layout area. During partitioning, the number of nets that are cut by the partition is usually minimized. In this case, the group migration method can be used.

5.1.4.1 Breuer's Algorithm

The main idea for Breuer's algorithm [20, 21] is to reduce the number of nets being cut when the circuit is partitioned. Various objective functions have been developed for this method. These objective functions are as given below.

1. **Total net-cut objective function:** All the nets that are cut by the partitioning are taken into account. This sum includes all nets cut by both horizontal and vertical partitioning cut lines. Minimizing this value is shown to be equivalent to minimizing the semi-perimeter wirelength [20, 21].

2. **Min-max cut value objective function:** In the case of standard cells and gate arrays, the channel width depends on the number of nets that are routed through the channel. The more the number of nets the larger

is the channel width and therefore the chip size. In this case the objective function is to reduce the number of nets cut by the cut line across the channel. This will reduce the congestion in channels having a large number of nets but at the expense of routing them through other channels that have a fewer number of nets or through the sparser areas of the channel.

3. **Sequential cut line objective function:** A third objective function is introduced to ease the computation of net cuts. Even though the above two objective functions represent a placement problem more accurately, it is very difficult to compute the minimum net cuts. This objective function reduces the number of nets cut in a sequential manner. After each partition, the number of nets cut is minimized. This greedy approach is easier to implement, however, it may not minimize the total number of nets cut.

In addition to the different objective functions, Breuer also presented several placement procedures in which different sequence of cut lines are used.

1. **Cut Oriented Min-Cut Placement:** Starting with the entire chip, the chip is first cut by a partition into two blocks. The circuit is also partitioned into two subcircuits so that the net cut is minimized. All the blocks formed by the partition are further partitioned by the second cut line and this process is carried out for all the cut lines. This partitioning procedure is sequential and easy to implement but it does not always yield good results because of the following two reasons. Firstly, while processing a cut line, the blocks created by the previous cut lines have to be partitioned simultaneously. Secondly, when a cut line partitions a block into two, the blocks to be placed in one of the partition might not fit in the partition created by the cut line as it might require more space than the block to be placed in the other partition (see Figure 5.7(a)).

2. **Quadrature Placement Procedure:** In this procedure, each region is partitioned into four regions of equal sizes by using horizontal and vertical cut lines alternatively. During each partitioning, the cutsize of the partition is minimized. As it cuts through the center and reduces the cutsize, this process reduces the routing density in the center. Currently, this is the most popular sequence of cut lines for min-cut algorithms (see Figure 5.7(b)).

3. **Bisection Placement Procedure:** The layout area is repeatedly bisected (partitioned into two equal parts) by horizontal cut lines until each subregion consists of one row. Each of these rows is then repeatedly bisected by vertical cut lines till each resulting subregion contains only one slot thus fixing the positions of all blocks. This method is usually used for standard cell placement and does not guarantee the minimization of the maximum net cut per channel (see Figure 5.7(c)).

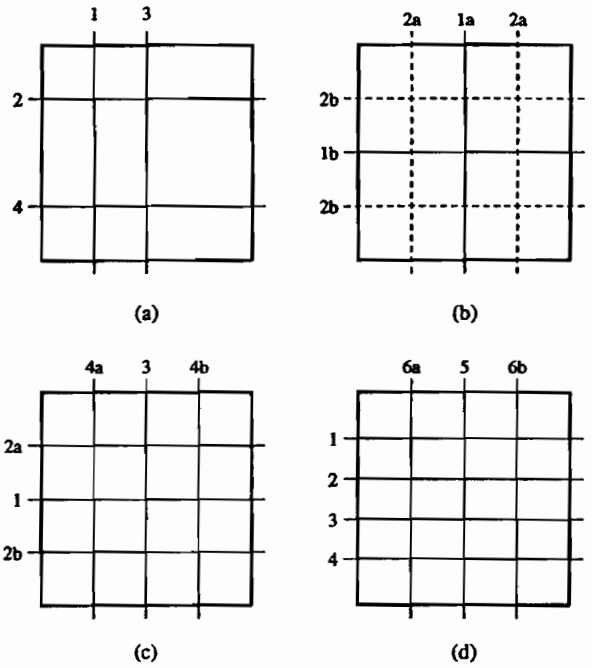


Figure 5.7: Different sequences of cut lines.

4. **Slice Bisection Placement Procedure:** In this method, a suitable number of blocks are partitioned from the rest of the circuit and assigned to a row, which is called a slicing, by horizontal cut lines. This process is repeated till each block is assigned to a row. The blocks in each row are then assigned to columns by bisecting using vertical cut lines. This technique is most suitable for circuits which have a high degree of interconnection at the periphery since this procedure tends to reduce the wire congestion at the periphery (see Figure 5.7(d)).

In any procedure described above, if the partitioning is to minimize the number of nets cut by the partition, a group migration method can be used in the partitioning process.

5.1.4.2 Terminal Propagation Algorithm

The partitioning algorithms partitioned the circuit merely to reduce the net cut. Therefore, the partitioning algorithms cannot be directly used for placement. This is illustrated in Figure 5.8. If the partitioning algorithm were to be used directly, terminals A and B may move away from each other as a result of

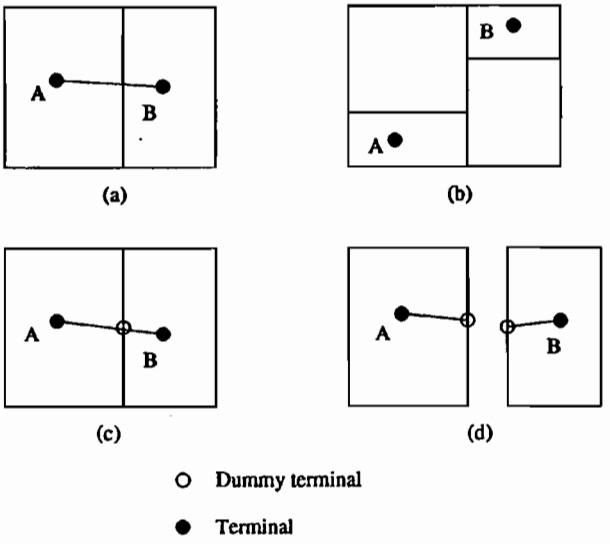


Figure 5.8: Terminal propagation.

partitioning, as shown in Figure 5.8(b). This not only increases the net length but increases the congestion in the channels as well. Hence unlike partitioning algorithms, placement algorithms which are based on partitioning need to preserve the information regarding the terminals which are connected and fall into two different partitions because of the cut. This can be done by propagating a dummy terminal to the nearest point on the boundary, when a net connecting two terminals is cut, as shown in Figure 5.8(c). When this dummy terminal is generated, the partitioning algorithm will not assign the two terminals in each partition, as shown in Figure 5.8(b), into different partitions as this would not result in a minimum cut. This method called the *terminal propagation* method was developed by Dunlop and Kerninghan [79].

5.1.5 Other Placement Algorithms

In this section, different kind of placement algorithms are described, which are neither simulation based nor partition based. These include cluster growth, quadratic assignment, resistive network optimization, and branch-and-bound algorithms.

5.1.5.1 Cluster Growth

In this constructive placement algorithm, the bottom-up approach is used. Blocks are placed sequentially in a partially completed layout. The seed or the

```
Algorithm CLUSTER-GROWTH
begin
    let B be the set of blocks to be placed;
    select a seed block B block from B;
    place B in the layout;
    B = B − B;
    while (B ≠ φ) do
        select a block B from B;
        place B in the layout;
        B = B − B;
end.
```

Figure 5.9: The Cluster growth algorithm.

first block is usually selected and placed by the user. After the seed block is placed, other blocks are selected and placed one by one to complete the layout. The selection and placement techniques differentiate in various cluster growth techniques.

In cluster growth algorithm, the block that is highly connected (have the most connections) to the already placed blocks is selected to be placed. Then, this block is placed either close to the block that it is highly connected to or a exhaustive search is carried out for the best possible location for the block. The outline of the cluster growth algorithm is shown in Figure 5.9.

The random constructive placement is a degenerate form of cluster growth. In this case, the selection of blocks is made randomly and its position is also fixed randomly. As this method does not take into account the interconnections and other circuit features, in most of the cases, it does not produce a good layout. This method is sometimes utilized to generate a basic layout for an iterative placement algorithm.

5.1.5.2 Quadratic Assignment

This method solves an abstract version of the gate array placement problem. It assumes that the blocks are points and have zero area. The cost of connecting two blocks B_i and B_j given by c_{ij} is stored in a connection matrix. The distance between slot k and slot l, given by d_{kl} is stored in a distance matrix. The objective is to map the blocks onto slots such that the product of connectivity and distance between the slots to which the blocks have been mapped (which gives the net length), for all the blocks, is minimized. This objective is equivalent to minimizing the total wire length for the circuit. This placement problem has been formulated as a quadratic assignment problem by Hall [131]. If C is the connection matrix and c_i is the sum of all elements in the ith row of C, then a diagonal matrix D can be defined as,

$$d_{ij} = \begin{cases} 0, & \text{if i} \neq \text{j}, \\ c_i, & \text{if } i = j \end{cases}$$

Let a matrix E be defined as $E = D - C$ and $X^T = [x_1, x_2, \ldots, x_n]$ and $Y^T = [y_1, y_2, \ldots, y_n]$ be row vectors representing the $x-$ and $y-$ coordinates of the desired solution. Hall proved that a nontrivial solution is obtained and the objective function is minimized if the smallest eigenvalues of the matrix E are chosen. The corresponding eigenvectors X and Y then give the coordinates of all the blocks.

5.1.5.3 Resistive Network Optimization

The placement problem has been transformed into the problem of minimizing the power dissipation in a resistive network by Cheng and Kuh [43]. The objective function, which is the squared Euclidean wire length, is written in a matrix form. This representation is similar to the matrix representation of resistive networks. This method can include fixed blocks in the formulation. Also, blocks with irregular sizes are allowed within cell rows. The algorithm comprises of subprograms which are used for optimization, scaling, relaxation, partitioning and assignment. The efficiency of the method comes from the fact that it takes advantage of the sparsity of the netlist. Slot constraints are used which guarantee the placement of blocks to be legal and each block is allocated to one slot. There are upto n constraints, where n is the number of blocks. The slot constraints are given by the equation

$$\sum_{i=1}^{n} x_i^j = \sum_{i=1}^{n} p_i^j, \qquad 1 \leq j \leq n$$

The algorithm maps the given circuit to a resistive network in which the pads and fixed blocks are represented as fixed voltage sources. Using the slot constraints the power dissipation in the circuit is minimized which causes the blocks to cluster around the center of the chip. The higher order slot constraints when applied cause the blocks to spread out. This step is called the *scaling step*. A repeated partitioning and relaxation then aligns the blocks with the slot locations.

5.1.5.4 Branch-and-Bound Technique

The general branch-and-bound algorithm can be applied to the placement problem. This method can be used for small circuits as it is a computationally intensive method. The method assumes that all the feasible solutions and the scores of these solutions are known. All these solutions make up a set called the *solution set*. The solution can be systematically searched. The search can be actually represented by a tree structure. The leaves of the tree are all the solutions. The selection of a solution is equivalent to traversing a branch of the tree and this step is called the *branch* step. If at any node in this tree a solution yields a score which is greater than the currently known lowest, then

the search continues in another part of the decision tree. This step is the *bound* step. Hence the algorithm actually prunes the decision tree which results in reduced computation.

Consider a gate array with three slots S_1, S_2, S_3 and three blocks B_1, B_2, B_3. At the first level of the tree, the root has three branches, each corresponding to a different placement of B_1 in three different slots. All the child nodes of the root will have two branches, each specifying two positions of B_2 in the remaining two slots. Finally, all grand children of root will have exactly one branch, specifying the slot for B_3.

The branch-and-bound algorithm traverses the tree and computes the cost of the solution at any given node. The cost can simply be the total wire length due to the placement of blocks upto that node. If this cost is higher than another known placement, this subtree need not be explored.

5.1.6 Performance Driven Placement

The delay at chip level, which depends on interconnecting wires plays a major role in determining the performance of the chip. As the blocks in a circuit become smaller and smaller, the size of the chip decreases. As a result, the delay due to the connecting wire becomes a major factor for high performance chips. The placement algorithms for high performance chips have to generate placements which will allow routers to route nets within the timing requirements. Such problems are called performance driven placement and the algorithms are called performance driven algorithms. The performance driven placement algorithms can be classified into two major categories, one which use the net-based approach and the other which use the path-based approach. In path-based approach [90, 170], the critical paths in the circuit are considered and the placement algorithms try to place the blocks in a manner that the path length is within its timing constraint. On the other hand, the net-based approach [63, 85, 91, 142, 225], tries to route the nets to meet the timing constraints on the individual nets instead of considering the paths. In this case, the timing requirement for each net has to be decided by the algorithm. Usually a pre-timing analysis generates the bounds on the netlengths which the placement algorithms have to satisfy while placing the blocks. Gao, Vaidya and Liu [105] presented a algorithm for high performance placement. The algorithm consists of the following steps:

1. Upper bounds for the netlengths are deduced from the timing requirements which is a part of the input to the algorithm. Each net has a set of such upper bounds. This provides the algorithm with maximum flexibility. The timing requirements are expressed by a set of linear constraints which are solved using convex programming techniques. A new convex programming algorithm is used for which the computational complexity depends only on the number of variables rather than the number of linear constraints.

2. A modified min-cut placement algorithm is used to obtain the placement of the blocks. The upper bounds calculated in the previous step guide the min-cut algorithm in placing the blocks. The min-cut algorithm, is a modified version of the Fiduccia's min-cut algorithm which tries to minimize the number of nets whose lengths exceed their corresponding upper bounds in addition to minimizing the size of the cutset.

3. The next step is to check whether all timing requirements are satisfied in the placement generated by the modified min-cut placement algorithm.

4. In case all the timing requirements are met, the placement is valid and is accepted otherwise the set of upper-bounds obtained in step 1 is modified and the steps 2 and 3 are repeated. Most other algorithms could not handle situations where the placement generated did not meet the timing specifications.

5.2 Floorplanning

Floorplanning is the placement of flexible blocks, that is, blocks with fixed area but unknown dimensions. It is a much more difficult problem as compared to the placement problem. In floorplanning, several layout alternatives for each block are considered. Usually, the blocks are assumed to be rectangular and the lengths and widths of these blocks are determined in addition to their locations. The blocks are assigned dimensions by making use of the aspect ratios. The aspect ratio of a block is the ratio of the width of the block to its length. Usually, there is an upper and a lower bound on the aspect ratio a block can have as the blocks cannot take shapes which are too long and very thin. Initial estimate on the set of feasible alternatives for a block can be made by statistical means, i.e., by estimating the expected area requirement of the block. Many techniques of general block placement have been adapted to floorplanning. The only difference between floorplanning and general block placement is the freedom of cells' interface characteristic. Like placement, inaccurate data partly affects floorplanning. In addition to the inaccuracy of the cost function that we optimize, the area requirements for the blocks may be inaccurate.

Floorplanning algorithms are typically used in hierarchical design. This is due to the fact that, although the dimensions of each leaf of the hierarchical tree may be known, the blocks at the node level in the tree are *flexible*, i.e., they can take any dimension. Hence, the floorplanning algorithms are used at each of the nodes in the tree so that the area of the layout is minimum and the position of all the blocks are identified.

5.2.1 Problem Formulation

As the placement problem is a restricted version of floorplanning all the constraints and objective functions applicable for the placement problem discussed

in the problem formulation for the placement problem are applicable. Usually, the input consists of $B_1, B_2, \ldots, B_n$ circuit blocks, with area $a_1, a_2, \ldots, a_n$ respectively. Associated with each block are two aspect ratios A_i^l and A_i^h, which give the lower and the upper bound on the aspect ratio for that block. The floorplanning algorithm has to determine the width w_i and height, h_i of each block B_i such that $A_i^l \leq \frac{h_i}{w_i} \leq A_i^h$. In addition to finding the shapes of the blocks, the floorplanning algorithm has to generate a valid placement such that the area of the layout is minimized.

A ***slicing floorplan*** is a floorplan which can be obtained by recursively partitioning a rectangle into two parts either by a vertical line or a horizontal line. The cut tree obtained by min-cut algorithm is known as ***slicing tree.*** A slicing tree is a binary tree in which each leaf represents a partition and each internal node represents a cut. Consider the floorplan as shown in Figure 5.10. Partitions are labeled with letters and cutlines are labeled with numbers. Figure 5.10(b) shows the slicing tree for the floorplan in Figure 5.10(a). Figure 5.10(c) is the slicing tree indicating the cut direction. Figure 5.10(d) shows a floorplan for which there is no valid slicing tree.

A floorplan is said to be *hierarchical* of *order* k, if it can be obtained by recursively partitioning a rectangle into at most k parts. The hierarchy of a hierarchical floorplan can be represented by a ***floorplan tree.*** Figure 5.11 shows a hierarchical floorplan of order 5 and its floorplan tree. Each leaf in the tree corresponds to a basic rectangle and each internal node corresponds to a composite rectangle in the floorplan. An important class of hierarchical floorplans is the set of all slicing floorplans.

5.2.1.1 Design Style Specific Floorplanning Problems

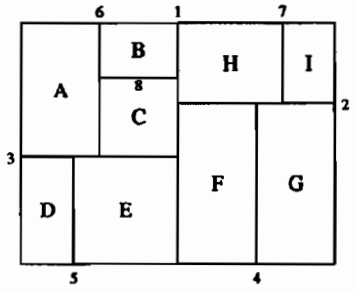
Floorplanning is not carried out for some design styles. This is due to the fixed dimensions of blocks in some design styles.

1. **Full custom design style:** Floorplanning for general cells is the same as discussed above.
2. **Standard cell design style:** In standard cell design style, the dimensions of cells are fixed, and floorplanning problem is simply the placement problem.
3. **Gate array design style:** Like standard cells, the floorplanning problem is same as placement problem.

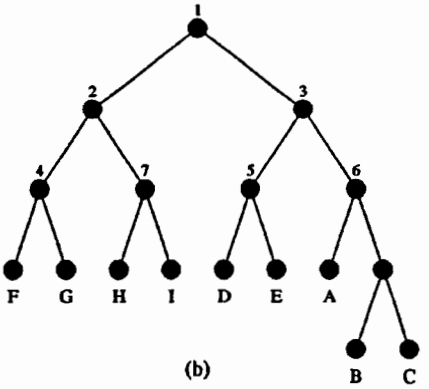
5.2.2 Classification of Floorplanning Algorithms

Floorplanning methods can be classified as follows:

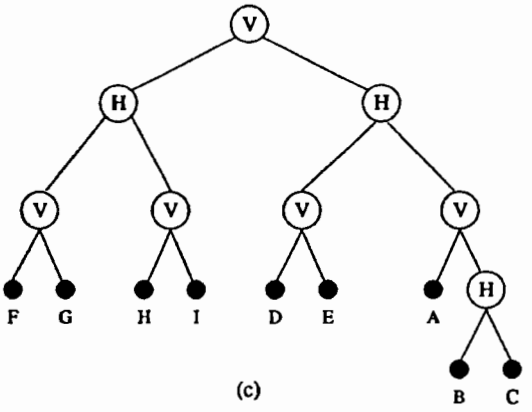
1. Constraint based methods.
2. Integer programming based methods.
3. Rectangular dualization based methods.



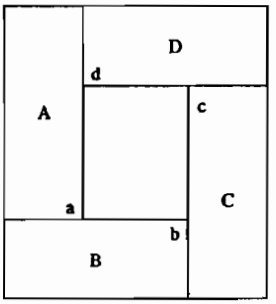
(a)



(b)



(c)



(d)

Figure 5.10: A floorplan with slicing tree and a non-slicing floorplan.

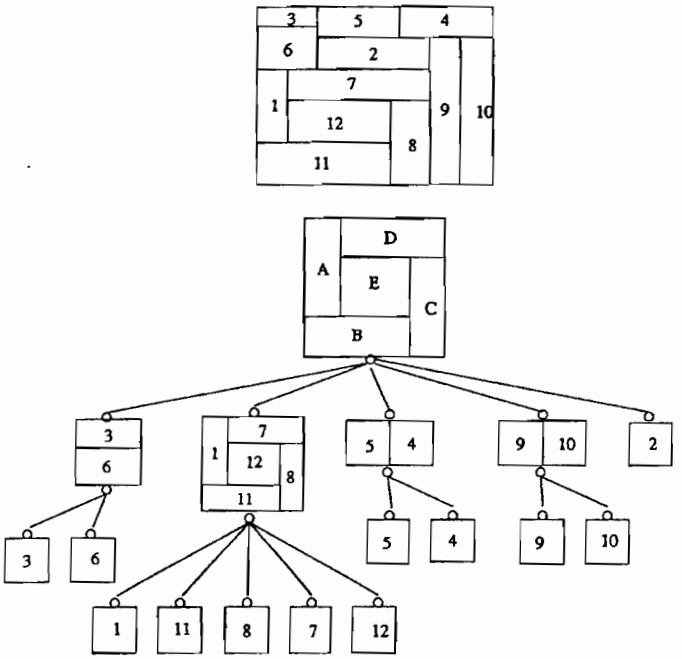


Figure 5.11: Hierarchical Floorplan.

These methods will be discussed in the following subsections.
Beside the methods stated above, several simple methods may be used, such as min-cut method. The process of min-cut can be used to construct a sized floorplan. The first phase of min-cut method i.e., bipartition of a weighted graph, helps in constructing the floorplan. The weight of the vertex roughly estimates the area taken up by the block. This weight may represent the area of the corresponding cell in general-cell placement. The initial sized floorplan represents an empty base rectangle whose area is the total of all weights of the vertices of the weighted graph and each node in the tree represents a rectangular room in the layout area. All the floorplans that can be generated with min-cut bipartitioning are slicing floorplans.

5.2.3 Constraint Based Floorplanning

This method, proposed by Vijayan and Tsay [334], constructs a floorplan of optimal area that satisfies (respects) a given set of constraints. A set of horizontal and vertical topological (i.e., ordering) constraints is derived from the relative placement of blocks. Given a constraint set, it is usually the case that there is no reason to satisfy all the constraints in the set. This is especially

true when a majority of the blocks have flexible shapes. A floorplan is said to respect a constraint, if for each pair of blocks, the floorplan satisfies at least one constraint (horizontal or vertical). A constraint set is said to be *overconstrained* if it has many redundant constraints. It is desirable to derive a complete constraint set from the input relative placement and then to remove those redundant constraints that result in reduction of floorplan area.

A topological constraint set of a set of blocks is given by two directed acyclic graphs (G_H, G_V): G_H is the horizontal constraint graph and G_V is the vertical constraint graph. In order to reduce the floorplan area, the heuristic iteratively removes a redundant constraint from the critical path of either G_H or G_V and also iteratively reshapes the blocks on the critical paths of the two graphs. Critical path is the longest path in G_H or G_V. The *input* to the algorithm is a constraint set (G_H, G_V) of the set of blocks. To minimize the floorplan area, repeat steps 1 and 2 until there is no improvement in the floorplan area.

1. **Step 1:** Repeat the following steps until are no strongly redundant edges on P_H or P_V exist. G_H and G_V are topologically sorted and swept. Either P_H or P_V, whichever is more critical is selected, where P_H and P_V are the critical paths of G_H and G_V respectively. The strongly redundant edge on the selected critical path is eliminated.

2. **Step 2:** The current shapes of the blocks are stored and a path, either P_V or P_H is selected depending on which of the two is more critical. All the flexible blocks on the selected path are reshaped. G_H and G_V are scanned again to construct the new floorplan. If the newly generated floorplan is better than the previous one the stored block shapes are updated. All the steps described above are repeated, a specified number of times.

Each pass of the algorithm constitutes one execution of two steps. Constraint reduction takes place in step 1 and step 2 does the reshaping of the blocks. If the chip dimensions are fixed, the passes are repeated until the target dimensions are reached. Otherwise the passes can be repeated until there is improvement in the floorplan area. Typically three or four passes are required.

The purpose of removing a redundant edge on the critical path is to break the path into two smaller paths. A good choice for such a redundant edge is the one which is nearest to the center point of the path. The above heuristic removes only one redundant constraint from a critical path at each iteration, and thus seeks to minimize the number of constraints removed. An edge can be checked for strong redundancy in constant time if we maintain the adjacency matrix of G_H and G_V. It takes $O(n^2)$ time to set up adjacency matrices. A topological sort of a directed acyclic graph with n nodes and m edges takes $O(m+n)$ time. The number of topological sorts executed depends on the number of redundant edges removed, the user-specified value for the number of reshaping iterations, and the number of passes.

5.2.4 Integer Programming Based Floorplanning

In this section an integer programming formulation for generating the floorplan developed by Sutanthavibul, Shragowitz and Rosen [314] is presented. The floorplanning problem is modeled as a set of linear equations using 0/1 integer variables. Two types of constraints are considered: the overlap constraints and the routability constraints. The overlap constraints prevent any two blocks from overlapping whereas the routability constraints estimate the routing area required between the blocks. For the critical nets, net lengths are specified which should not be exceeded. The length of the net depends on the timing budget of that net. The critical net constraints ensure that the length of the critical nets does not exceed this specified value. We now describe how the constraints can be developed.

1. **Block overlap constraints for fixed blocks:** Given two fixed (rigid) blocks, B_{ri} and B_{rj} which should not overlap, we have four possible ways to position the two blocks so as to avoid overlap. Let $\{x_i, y_i, w_i, h_i\}$ and $\{x_j, y_j, w_j, h_j\}$ be the 4-tuples associated with blocks B_{ri} and B_{rj} respectively, where (x_i, y_i) gives the location of the block, w_i is the width of the block and h_i is the height of the block. The block B_{rj} can be positioned to the right, left, above or below block B_{ri}. These conditions transformed into equations given below:

$$\begin{array}{lll} x_i + w_i \leq x_j & (B_{rj} \text{ is to the right of } B_{ri}), & \text{or} \\ x_i - w_j \geq x_j & (B_{rj} \text{ is to the left of } B_{ri}), & \text{or} \\ y_i + h_i \leq y_j & (B_{rj} \text{ is to the above of } B_{ri}), & \text{or} \\ y_i - h_j \geq y_j & (B_{rj} \text{ is to the below of } B_{ri}) & \end{array} \quad (1)$$

To satisfy one of these equations, two 0-1 integer variables x_{ij} and y_{ij} are used for each pair of blocks. Two bounding functions W and H are defined such that, $|x_i - x_j| \leq W$ and $|y_i - y_j| \leq H$. W can be equal to $W_{\max}$ which is the maximal allowed width of the chip or $W = \sum_{i=1}^{p+q+r} w_i$. Similarly, $H = H_{\max}$, the maximal allowed height of the chip or $H = \sum_{i=1}^{p+q+r} h_i$. Equation set (1) can be rewritten with the introduction of the integer variables to generate the 'or' condition as,

$$\begin{aligned} & x_i + w_i \leq x_j + W(x_{ij} + y_{ij}) \\ & x_i - w_j \geq x_j + W(1 - x_{ij} + y_{ij}) \\ & y_i + h_i \leq y_j + W(1 + x_{ij} - y_{ij}) \\ & y_i - h_j \geq y_j + W(2 - x_{ij} - y_{ij}) \end{aligned} \quad (2)$$

As the integer variables x_{ij} and y_{ij} can take either 0 or 1 values, only one of the above equations in (2) will be active and other equations will be true depending on the value of x_{ij} and y_{ij}. For example, when $x_{ij} = y_{ij} = 1$, the first equation in (2) becomes active and all other equations are true.

$$\begin{aligned} & x_i \geq 0, \qquad y_i \geq 0 \\ & x_i + w_i \leq W, \\ & y^* \geq y_i + h_i \end{aligned} \quad (3)$$

where y^* is the height to be minimized. To allow rotation of the blocks so as to optimize the solution, another integer variable z_i is used for each block. z_i is 0 when the block is in its initial orientation and 1 when the block is rotated by 90°. The constraints for the fixed blocks can be rewritten as:

$$\begin{aligned} &x_i + z_i h_i + (1 - z_i) w_i \leq x_j + M(x_{ij} + y_{ij}) \\ &x_i - z_i h_j - (1 - z_j) w_j \geq x_j + M(1 - x_{ij} + y_{ij}) \\ &y_i + z_i w_i + (1 - z_i) h_i \leq y_j + M(1 + x_{ij} - y_{ij}) \\ &y_i - z_j w_j - (1 - z_j) h_j \geq y_j + M(2 - x_{ij} - y_{ij}) \end{aligned} \tag{4}$$

where, $M = \max(W, H)$. Constraints (3) are rewritten as:

$$\begin{aligned} &x_i \geq 0, \quad y_i \geq 0, \\ &x_i + (1 - z_i) w_i + z_i h_i \leq W, \\ &y^* \geq y_i + (1 - z_i) w_i + z_i h_i \end{aligned} \tag{5}$$

where y^* is the height to be minimized. The floorplanning problem, for fixed blocks without taking into consideration either routing areas or critical nets can be solved by finding the minimum y^* subject to constraints (4) and (5).

2. **Block overlap constraints for flexible blocks:** So far we discussed about fixed blocks. We can now see how constraints for flexible blocks can be developed. The flexible blocks can take rectangular shapes within a limited aspect ratio range i.e. its width and height can be varied keeping the area fixed. The non-linear area relation is linearized about the point of maximum allowable width by applying the first two members of the Taylor series giving,

$$h_i = h_{i0} + \Delta w_i \lambda_i$$

where,

$$\begin{aligned} h_{i0} &= \frac{A_i}{w_{imax}}, \\ \lambda_i &= \frac{A_i}{w_{imax}^2}, \\ \Delta w_i &= w_{imax} - w_i \end{aligned}$$

where Δw_i is a continuous variable for block B_{fi}. The overlap constraints for a flexible block B_{fi} and a fixed block B_{rj} can be written as:

$$\begin{aligned} &x_i + w_{imax} - \Delta w_i \leq x_j, && (B_{fj} \text{ is to the right of } B_{ri}), && \text{or} \\ &y_i + h_{i0} + \Delta w_i \lambda_i \leq y_j, && (B_{fj} \text{ is above } B_{ri}), && \text{or} \\ &x_i - w_j \geq x_j, && (B_{fj} \text{ is to the left of } B_{ri}), && \text{or} \\ &y_i - h_j \geq y_j, && (B_{fj} \text{ is below } B_{ri}) \end{aligned} \tag{6}$$

Using two integer variables x_{ij} and y_{ij} per block pair as was done for fixed blocks, the 'or' condition between the equations can be satisfied.

The same set of equations can be extended to get overlap constraints between two flexible blocks. Using the same technique, the interconnection length constraints and routing area constraints can be developed. This set of equations are the input to any standard linear programming software package such as LINDO. The locations of the blocks and their dimensions are variables, the values of which are calculated by the software depending on the constraints and the objective function.

5.2.5 Rectangular Dualization

The partitioning process generates a group of subcircuits and their interconnections. This output from a partitioning algorithm can be represented as a graph $G = (V, E)$ where the vertices of the graph correspond to the subcircuits and the edges represent the interconnections between the subcircuit. The floorplan can be obtained by converting this graph into its *rectangular dual* and this approach to floorplanning is called *rectangular dualization.* A rectangular dual of graph $G = (V, E)$ consists of non-overlapping rectangles which satisfy the following properties:

1. Each vertex $v_i \in V$ corresponds to a distinct rectangle R_i, $1 \leq i \leq |V|$.
2. For every edge $(v_i, v_j) \in E$, the corresponding rectangles R_i and R_j are adjacent in the rectangular dual.

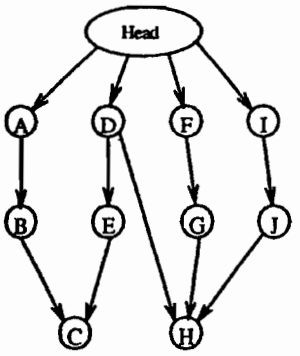
When this method is directly applied to the graph generated by partitioning, it may not be possible to satisfy the second property for generating the rectangular dual.

The problem of finding a suitable rectangular dual is a hard problem. In addition, there are many graphs which do not have rectangular duals. A further complication arises due to areas and aspect ratios of the blocks. In rectangular dualization, areas and aspect ratios are ignored to simplify the problem. As a result, the output cannot be directly used for floorplanning. Kozminski and Kinnen [177] have presented an algorithm for finding a rectangular dual of a planar triangulated graph. Usually, the graph is processed only if a rectangular dual for the graph exists. Bhasker and Sahni [11] have extended the approach in [177] to present a linear time algorithm for finding a rectangular dual of a planar triangulated graph.

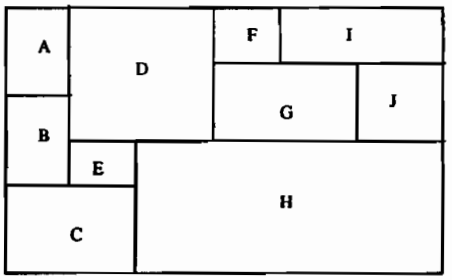
A planar triangular graph (PTG) G is a connected planar graph that satisfies the following properties:

1. every face (except the exterior) is a triangle.
2. all the internal vertices have a degree ≥ 4.
3. all cycles that are not faces have length ≥ 4.

Given a PTG, a planar digraph is constructed which is a directed graph. Once a planar digraph is constructed, it can be converted into a floorplan as shown



(a)



(b)

Figure 5.12: Conversion of planar digraph to a floorplan.

in Figure 5.12. Lokanathan and Kinnen [216] presented a procedure for floorplanning that minimizes routing parasitics using rectangular dualization. The use of rectangular dualization maximizes adjacency of blocks that are heavily connected or connected by critical nets.

5.3 Pin Assignment

The purpose of pin assignment is to define the signal that each pin will receive. Pin assignment may be done during floorplanning, placement or after placement is fixed. If the blocks are not designed then good assignment of nets to pins can improve the placement. If the blocks are already designed, it may be possible to exchange a few pins. This is because some pins are *functionally equivalent* and some are *equipotential*. Two pins are called functionally

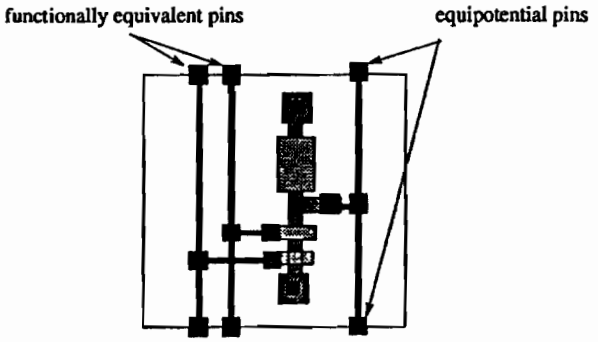


Figure 5.13: Functionally equivalent and equipotential pins.

equivalent, if exchanging the signals does not effect the circuit. For example, exchanging two inputs of a gate does not effect the output of the gate. Two pins are equipotential if both are internally connected and hence represent the same net. The output of the gate may be available on both sides, so the output signal can be connected on any side. Figure 5.13 shows both functionally equivalent pins and equipotential pins.

5.3.1 Problem Formulation

The purpose of pin assignment is to optimize the assignments of nets within a functionally equivalent pin groups or assignment of nets within an equipotential pin group. The objective of pin assignment is to reduce congestion or reduce the number of crossovers. Figure 5.14 illustrates the effectiveness of pin assignment. Note that a net can be assigned to any equipotential pin within a set of functionally equivalent pins. The pin assignment problem can be formally stated as follows: Given a set of terminals $T_1, T_2, \ldots, T_n$ and a set of pins $P_1, P_2, \ldots, P_m$, $m > n$. Each T_i is assigned to pin P_i, $i = 1, 2, \ldots, n$. Let $\mathcal{E}_{P_i}$ be the set of pins which are equipotential and equivalent to P_i, the objective of pin assignment is to assign each T_i to a pin in $\mathcal{E}_{p_i}$ such that a specific objective function is minimized. The objective functions are typically routing congestions. For standard cell design, it may be the channel density.

5.3.1.1 Design Style Specific Pin Assignment Problems

Pin assignment problems in different design styles have different objectives.

1. **Full custom design style:** In full custom, we have two types of pin assignment problems. At floorplanning level, the pin location along the boundary of the block can be changed as the block is assigned a shape. This assignment of pins can reduce routing congestions. Thus, not only

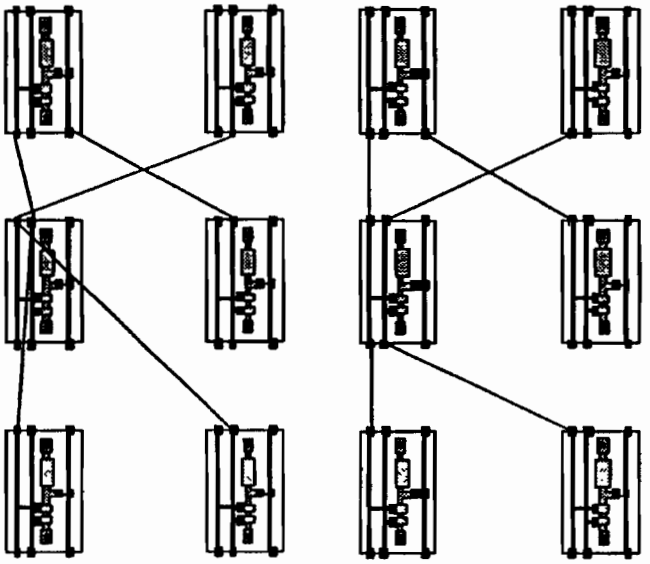


Figure 5.14: Impact of pin assignment.

we can change pin assignment of pins, we can also change the location of pins along the boundary. At placement level, the options are limited to assigning the nets to pins. Notice that in terms of problem formulation, we can declare all pins of a flexible block as functionally equivalent to achieve pin assignment in floorplanning.

2. **Standard cell design style:** The pin assignment problem for standard cells is essentially that of permuting net assignment for functionally equivalent pins or switching equipotential pins for a net.

3. **Gate array design style:** The pin assignment problem for gate array design style is the same as that of standard cells.

Assignment problems mostly occur in semi custom design styles such as gate arrays or standard cells.

In gate array design, the cells are pre-fabricated and are arranged on the master. Pin assignment problem in this type of design style is to assign to each terminal a functionally equivalent slot such that wiring cost is minimized. Slots in this case are the pin locations on pre-designed (library) cells. In standard cells, however, equipotential pins appear as feedthroughs. Since no wiring around the cell is needed, the wire length decreases with the use of feedthroughs.

5.3.2 Classification of Pin Assignment Algorithms

The pin assignment techniques are classified into general techniques and special pin assignment techniques. General techniques are applicable for pin assignment at any level and any region within a chips. Such techniques are applied at board level as well as chip level. On the other hand, the special pin assignment technique can be used for assignment of pins within a specific region such as channel or a switchbox.

5.3.3 General Pin Assignment

There are several methods in this category as discussed below:

1. **Concentric Circle Mapping:** To planarize the interconnections, this method models the pin assignment problem by using two concentric circles [194]. The pins on the component being considered are represented as points on the inner circle whereas the points on the outer circle represent the interconnections to be made with other components. The concentric circle mapping technique solves the pin assignment problem by breaking it into two parts. The first part is the assignment of pins to points on the two circles and in the second part the points on the inner and outer circles are mapped to give the interconnections.

   For example, consider the component and the pins shown in Figure 5.15(a). The two circles are drawn so that the inner circle is inside all the pins on the component being considered while the outer circle is just inside the pins that are to be connected with the pins of this component. This is shown in Figure 5.15(b). Lines are drawn from the component center to all these pins as shown in Figure 5.15(c). The points on the inner and outer circle are defined by the intersection of these lines with the circles (Figure 5.15(d)). The pin assignment is completed by mapping the points on the outer circle to those on the inner circle in a cyclic fashion. The worst and the best case assignment are shown in Figure 5.15(e) and (f).

2. **Topological Pin Assignment:** Brady [18] developed a technique which is similar to concentric circle mapping and has certain advantages over the concentric circle mapping method. With this method it is easier to complete pin assignment when there is interference from other components and barriers and for nets connected to more than two pins. If a net has been assigned to more two pins than the pin closest to the center of the primary component is chosen and all other pins are not considered. Hence in this case only one pin external to the primary component is chosen. The pins of the primary component are mapped onto a circle as in the concentric circle method. Then beginning at the bottom of the circle and moving clockwise the pins are assigned to nets and hence they get assigned in the order in which the external pins are encountered. For nets with two pins the result is the same as that for concentric circle mapping.

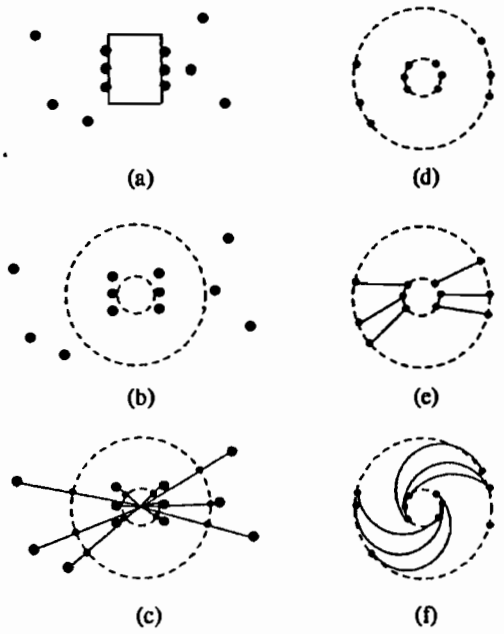


Figure 5.15: Concentric circle mapping.

1978

3. **Nine Zone Method:** The nine zone method, developed by Mory-Rauch, is a pin assignment technique based on zones in a Cartesian coordinate system [237]. The center of the coordinate system is located inside a group of interchangeable pins on a component. This component is called *pin class*. A net rectangle is defined by each of the nets connected to the pin class. There are nine zones in which this rectangle can be positioned as shown in Figure 5.16. The positions of these net rectangles are defined relative to the coordinate system defined by the current pin class.

5.3.4 Channel Pin Assignment

In design of VLSI circuits, a significant portion of the chip area is used for channel routing. Usually, after the placement phase, the positions of the terminals on the boundaries of the blocks are not completely fixed and they still have some degree of freedom to move before the routing phase begins. Figure 5.17 shows how channel density could be reduced by moving the terminals. Figure 5.17(a) shows a channel which needs three tracks. By moving the pins, the routing can be improved such that it requires one track as shown in Figure 5.17 (b). The channel pin assignment problem is the problem of assigning positions for the terminals, subject to constraints imposed by design rules and

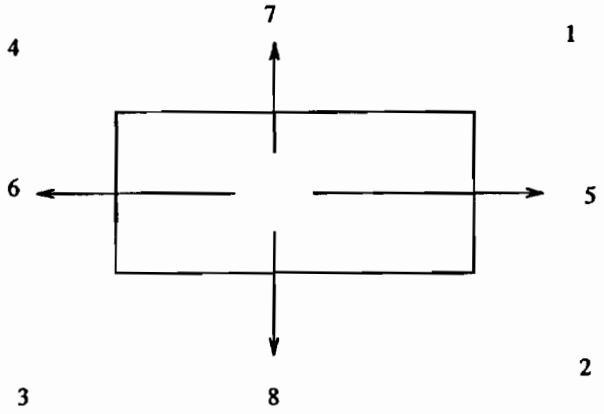


Figure 5.16: The nine pin zones.

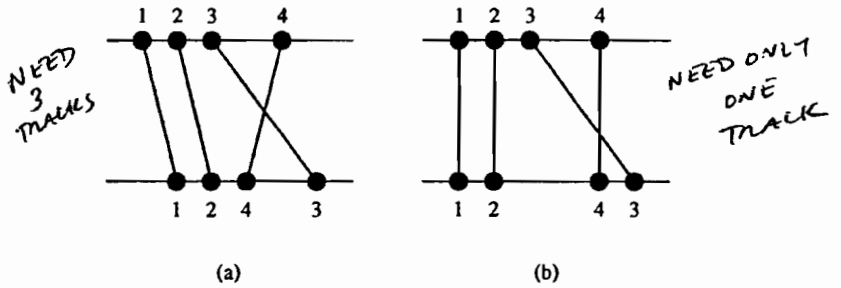


Figure 5.17: Reducing channel density by moving terminals.

the designs of the previous phases, so as to minimize the density of the channel. The problem has various versions depending on how the pin assignment constraints are specified. Many special cases of this problems have been investigated. In [120], Gopal, Coppersmith, and Wong considered the channel routing problem with movable terminals.

In [347] the channel pin assignment problem in which assignment of terminals is subject to linear order position constraints is solved using a dynamic programming formulation by Yang and Wong. Their method is described briefly below. Except minor changes for clarity, the discussion is essentially same as it appears in [347]. 1991

Since the terminals are linearly ordered we have a set of terminals at the top given by TOP in which the terminals $t_1 < t_2 < \ldots < t_p$. Similarly the terminal set for terminals at the bottom is given by BOT in which the terminals are

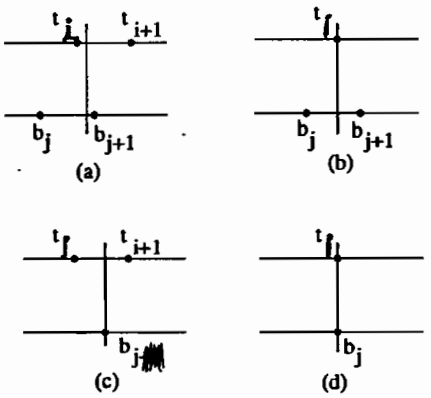


Figure 5.18: Four types of (i, j, k) solutions.

ordered $b_1 < b_2 < \ldots < b_q$. Each terminal t_i on the top and b_i on the bottom have a corresponding set given by T_i and B_i which indicate the possible positions these terminals can occupy. A solution to this problem is called an *(i,j,k)-solution* if it assigns exactly $t_0, t_1, t_2, \ldots, t_i, b_0, b_1, b_2, \ldots, b_j$ to the first k columns of the channel where t_0 and b_0 correspond to a auxiliary column and two trivial nets which consist of only one terminal are introduced. The main idea used by the algorithm is to first compute a density function using dynamic programming and then use backtracking to reconstruct an optimal solution.

Let L be the length of the channel and N be the number of nets to be routed. The set of terminals on the top boundary of the channel is denoted as TOP, and the set of terminals on the bottom boundary of the channel is denoted as BOT. In this implementation, separation constraints and position constraints are not considered and but the terminals are definitely allowed to be within a certain position only i.e. the length L of the channel. The (i, j, k) solutions can be classified into the following four types, according to pin assignment at column k as illustrated in Figure 5.18.

- Type 0: No terminal is assigned to either endpoints of column k as shown in Figure 5.18(a).
- Type 1: Only t_i is assigned to the top endpoint of column k as shown in Figure 5.18(b).
- Type 2: Only b_j is assigned to the bottom endpoint of column k as shown in Figure 5.18(c).
- Type 3: Both t_i and b_j are assigned to column k as shown in Figure 5.18(d).

Let $d(i,j,k)$ be the density of the channel considering i terminals at the top and j terminals at the bottom after consideration of k columns. Let $x(i,j,k)$, $y(i,j,k)$ and $z(i,j,k)$ be the local densities (*crossing numbers*) at column k considering i nets at the top and j nets at the bottom for Type 1, Type 2 and Type 3 solutions respectively. Let $R_1(i,j)$ denote the set of nets with one terminal in $\{t_1,t_2,\ldots,t_{i-1},b_1,b_2,\ldots,b_j\}$ and one terminal in TOP ∪ BOT - $\{t_1,t_2,\ldots,t_i,b_1,b_2,\ldots,b_j\}$, and the net containing t_i, if it is not trivial. Let $R_2(i,j)$ denote the set of nets with one terminal in $\{t_1,t_2,\ldots,t_i,b_1,b_2,\ldots,b_{j-1}\}$, and one terminal in TOP ∪ BOT - $\{t_1, t_2, \ldots, t_i, b_1, b_2, \ldots, b_j\}$, and the net containing b_j, if it is not trivial. Let $R_3(i,j)$ denote the set of nes with one terminal in $\{t_1,t_2,\ldots,t_{i-1},b_1,b_2,\ldots,b_{j-1}\}$, and one terminal in TOP ∪ BOT -$\{t_1,t_2,\ldots,t_i,b_1,b_2,\ldots,b_j\}$, and the net containing t_i or b_j, if it is not trivial and if they do not belong to the same net. Hence we have

$$x(i,j,k) = \begin{cases} +\infty & \text{if } k \notin T_i \\ |R_1(i,j)| & \text{otherwise} \end{cases}$$

$$y(i,j,k) = \begin{cases} +\infty & \text{if } k \notin B_j \\ |R_2(i,j)| & \text{otherwise} \end{cases}$$

$$z(i,j,k) = \begin{cases} +\infty & \text{if } k \notin T_i \cap B_j \\ |R_3(i,j)| & \text{otherwise} \end{cases}$$

$|R_1(i,j)|$ = # OF NETS THAT CROSS COLUMN k.

The algorithm for optimal channel pin assignment is shown in Figure 5.19. It is easy to see that the time complexity of the algorithm Linear-CPA is $O(pql)$.

5.4 Integrated Approach

The various stages in the physical design cycle evolved as the entire problem is extremely complex to be solved altogether at once. But over the years, with better understanding of the problems, attempts are made to merge some steps of the design cycle. For example, floorplanning was considered as a problem of just finding the shapes of the blocks without considering routing areas. Over the years, the floorplanning problem has been combined with the placement problem [63, 314, 339]. The placement problem is sometimes combined with the routing problem giving rise to the 'place and route' algorithms [84, 100, 292, 316].

In this section, the approach used by Dai, Eschermann, Kuh and Pedram in BEAR [63] is described briefly. BEAR is a macrocell-based layout system. The process of floorplanning is carried out in the following three steps:

1. **Clustering:** In this step, a hierarchical tree is constructed. Blocks that are strongly connected are grouped together in a cluster. Each cluster can have a limited number of blocks within it. The clustering process considers the shapes of the blocks to avoid a mismatch within the cluster. This step is repeated to build the cluster tree.

$\Rightarrow O(pqL) \Rightarrow O(nmk)$ FROM CLASS NOTES

p: # TERMS ON TOP
q: # TERMS ON BOTTOM

```
Algorithm LINEAR-CPA()
begin
    (* initialize *)
    for i = 0 to p do
        for j = 0 to q do
            d(i, j, 0) = +∞;
    for k = 0 to L do
        d(0, 0, k) = 0;
    COMPUTE-CROSS();
    for k = 1 to L do
        for i = 0 to p do
            for j = 0 to q do
                (* type 1 solution *)
                D1 = max{d(i − 1, j, k − 1), x(i, j, k)};
                (* type 2 solution *)
                D2 = max{d(i, j − 1, k − 1), y(i, j, k)};
                (* type 3 solution *)
                D3 = max{d(i − 1, j − 1, k − 1), z(i, j, k)};

                d(i, j, k) = min{d(i, j, k − 1), D1, D2, D3};
    if d(p, q, l) = +∞ then
        return φ is not feasible;
    else
        (* backtracking for constructing optimal solution *)
        i = p; j = q;
        for k = L down to 1 do
            if d(i, j, k) = D1 then
                f(i) = k; i = i − 1;
            else if d(i, j, k) = D2 then
                g(j) = k; j = j − 1;
            else if d(i, j, k) = D3 then
                f(i) = k; g(j) = k;
                i = i − 1; j = j − 1;
        return π = (f, g);
end.
```

Figure 5.19: The optimal channel pin assignment algorithm

```
Procedure COMPUTE-CROSS()
begin
    for (i = 0 to p)do
        for (j = 0 to q) do
            COMPUTE( R1(i,j) );   (* Compute |R1(i,j)| *)
            COMPUTE( R2(i,j) );   (* Compute |R2(i,j)| *)
            COMPUTE( R3(i,j) );   (* Compute |R3(i,j)| *)
            for (k = 0 to L) do
                x(i,j,k) = +∞;
                y(i,j,k) = +∞;
                z(i,j,k) = +∞;
    for(i = 0 to p) do
        for(j = 0 to q)do
            for( k ∈ Ti ) do
                x(i,j,k) = |R1(i,j)|;
            for( k ∈ Bj ) do
                y(i,j,k) = |R2(i,j)|;
            for( k ∈ Ti ∩ Bj ) do
                z(i,j,k) = |R3(i,j)|;
end.
```

Figure 5.20: The COMPUTE-CROSS procedure for the channel pin assignment algorithm

2. **Placement:** In this step, the tree is traversed top-down. The target shape and terminal goals for the root of the cluster tree is specified. This information is used to identify the topological possibility for the clusters at the level below. This in turn sets the shape and terminal goals for the immediate lower levels in the hierarchy till at the leaf level the orientations of the blocks are determined. For each of the topologies, the routing space is determined. The selection of a particular topology is based on the area and the shape of the resulting topology and the connection cost. The system is developed so as to allow the user to control the trade off between the shape, the area and the connection costs. This strategy works well in case the blocks at the leaf level are flexible so that the shapes of these blocks can be adjusted to the shape of the cluster. On the other hand, if the leaf level blocks are fixed then this top-down approach can give unfavorable results. This is due to the fact that the information of the shape of these blocks at the leaf level are not considered by the objective function when determining the cluster shapes at higher levels of the cluster tree. This is rectified by passing the shape information from the leaves towards the root of the tree during the clustering phase. In addition, during the top-down placement step, a look-ahead is added so that the objective function can examine the

shapes generated during clustering, at a level below the immediate level for which the shape is being determined.

3. **Floorplan optimization:** This an improvement step that resizes selected blocks iteratively. The blocks to be resized are identified by computing the longest path through the layout surface using the routing estimates done in the previous step.

5.5 Summary

Placement and floorplanning are key steps in physical design cycle. Floorplanning is the super problem of placement, since in addition to finding the location of blocks, it finds appropriate shapes for each block. The pin assignment is usually carried out after the blocks have been placed to reduce the complexity of the overall problem.

Several placement algorithms have been presented. Simulated annealing and simulated evolution are two most successful placement algorithm. Although these algorithms are computationally intensive, they do produce good placements. Integer programming based algorithms for floorplanning have been also been successful. Several algorithms have been presented for pin assignment, including optimal pin assignment for channel pin assignment problems. The output of the placement phase must be routable, otherwise placement has to be repeated.

5.6 Exercises

1. Given the following 14 rectangles with their dimensions specified, write a program that will arrange all these rectangles within 5000sq. units of area, if possible, or otherwise minimize the area required. The dimensions (width × height) of the rectangles are $R_1 = 15 \times 15, R_2 = 25 \times 15, R_3 = 10 \times 30, R_4 = 30 \times 20, R_5 = 10 \times 15, R_6 = 20 \times 5, R_7 = 10 \times 25, R_8 = 30 \times 15, R_9 = 10 \times 65, R_{10} = 10 \times 25, R_{11} = 20 \times 20, R_{12} = 10 \times 20, R_{13} = 30 \times 15, R_{14} = 40 \times 15$.

2. Recall that the aspect ratio of a block is the ratio of it s height and width. If each rectangle in problem 1 can have three different aspect ratios, find the appropriate aspect ratio for each rectangle so that the area occupied by the rectangles is minimized. The set of aspect ratios for R_i, is R_i^a, is, $R_1^a = \{1.0, 1.2, 2.0\}$, $R_2^a = \{0.8, 1.5, 1.9\}$, $R_3^a = \{0.6, 2.0, 3.0\}$, $R_4^a = \{0.8, 1.2, 1.5\}$, $R_5^a = \{0.75, 1.2, 1.5\}$, $R_6^a = \{0.3, 0.9, 2.0\}$, $R_7^a = \{0.4, 1.2, 1.5\}$, $R_8^a = \{0.5, 1.0, 1.5\}$, $R_9^a = \{0.8, 3.0, 4.0\}$, $R_{10}^a = \{0.4, 1.2, 1.8\}$, $R_{11}^a = \{0.5, 1.0, 1.2\}$, $R_{12}^a = \{0.5, 0.9, 1.2\}$, $R_{13}^a = \{0.5, 1.0, 1, 5\}$, $R_{14}^a = \{0.4, 1.6, 2, 5\}$.

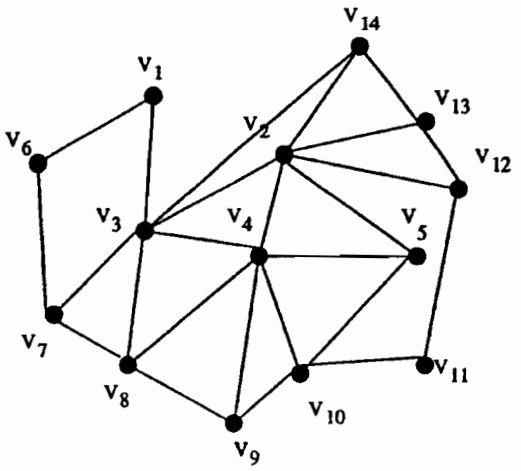


Figure 5.21: Example of a connectivity graph.

3. Use the lowest and highest aspect ratios for each rectangle in problem 2 as lower and upper bounds respectively and generate a placement which occupies minimum area.

4. Consider the first 10 blocks given in problem 1. If D_{ij} represents the center to center distance between blocks i and j then determine if these blocks can be placed together so that the distances between the blocks are within their specified values. The distances that are to be maintained are $D_{12} = 30, D_{23} = 35, D_{14} = 18, D_{34} = 40, D_{24} = 30, D_{13} = 45$.

† 5. Implement the Simulated Annealing algorithm. Consider the graph shown in Figure 5.21. Each vertex represents rectangle R_i whose dimensions are specified in problem 1, the edges of the graph represent the connectivity of the blocks. Use the Simulated Annealing algorithm to generate a placement.

† 6. For the placement obtained in problem 5, implement a pin assignment algorithm which will reduce the total net length and minimize the maximum length of a net. Generate a complete routing for this placement. The routing can be generated on an uniform grid and two nets can intersect only when they are perpendicular to each other.

‡ 7. Implement the Simulated annealing algorithm for the general cell placement problem.
Hint: Instead of exchanging a big block with a small block, a big block can be exchanged with a cluster of small blocks.

‡ 8. For a given placement, implement a pin assignment algorithm which will rotate the pins on the blocks either in clockwise or anticlockwise direction till the total net length is reduced. While rotating the pins on the blocks, the order of these pins must be maintained.

9. For the blocks specified in problem 1 generate the integer program constraints to solve the placement problem. Consider some of the blocks as flexible and solve the floorplanning problem using the integer program.

† 10. Implement a placement algorithm for high performance circuits which takes into account path delays instead of net delays.

‡ 11. Modify the min-cut algorithm to incorporate the terminal propagation scheme.

‡ 12. Develop Simulated Evolution algorithm for standard cells design.

‡ 13. Develop Simulated Evolution algorithm for full custom designs.

‡ 14. Several industrial libraries allow cells with different cell heights. This leads to irregular shape channels. Suggest modifications required for applying the Simulated annealing algorithm to standard cells of uneven heights.

† 15. Implement force directed placement algorithm for gate array design style.

† 16. Apply Simulated Annealing algorithm for pin assignment problem. In each iteration, pins of each blocks are permuted and routing congestion is estimated.

† 17. Develop an algorithm for pin assignment of a full custom layout based on concentric circle mapping.

‡ 18. Implement the channel pin assignment algorithm. Discuss the constraints, based on functionally equivalent and equipotential pins.

Bibliographic Notes

A linear assignment algorithm for the placement problem has been discussed in [3]. Two partition/interchange processes are described in [251] for solving the placement problem. The graph is partitioned into several smaller graphs for initial placement in both the methods and finally interchange optimization is carried out. The simulated annealing optimization method has been adapted to the placement of macros on chips for full custom design in [172]. A timing driven placement system has been discussed in [76]. A hierarchical placement procedure incorporating detailed routing and timing information has been discussed in [106]. The procedure is based on the min-cut method. Global routing and timing analysis is carried out after every cut which guides the subsequent cell partitioning. [203] discusses an interactive program to get a good floorplan. It includes graphical output, block and pad manipulation and a cost function for estimation of total wire length. A floorplanning system

designed to work within a hierarchical design environment supporting multiple design styles has been discussed in [229]. A technique for floorplanning and pin assignment of general cell layouts has been developed in [252]. A global floorplanning approach has been discussed in [257]. The approach is based on a combined min-cut and slicing paradigm. A pin assignment algorithm for improving the performance in standard cell design by improving the longest delay has been discussed in [284]. A pin assignment problem for macro-cells is discussed in [350]. A approach which combines pin assignment and global routing has been developed in [51].

Bibliography

[1] I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga, "An Implementation of Karmarkar's Algorithm for Linear Programming," *Math. Program.*, 44, pp. 297–335, 1989.

[2] S. B. Aker, "A Modification of Lee's Path Connection Algorithm," *IEEE Transactions on Computers*, pp. 97–98, February 1967.

[3] S. B. Akers, "On the Use of the Linear Assignment Algorithm in Module Placement," *Proceedings of 18th ACM/IEEE Design Automation Conference*, pp. 137–144, 1981.

[4] S. B. Akers, J. M. Geyer, and D. L. Roberts, "IC Mask Layout with a Single Conductor Layer," *Proceedings of 7th Design Automation Workshop*, pp. 7–16, 1970.

[5] K. J. Antreich, F. M. Johannes, and F. H. Kirsch, "A New Approach for Solving the Placement Problem Using Force Models," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 481–486, 1982.

[6] P. B. Arnold, "Complexity Results for Circuit Layout on Double-sided Printed Circuit Boards," Undergraduate thesis, Department of Applied Mathematics, Harvard University, May 1982.

[7] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison Wesley, 1990.

[8] P. Bannerjee, and M. Jones, "A Parallel Simulated Annealing Algorithm for Standard Cell Placement on A Hypercube Computer," *Proceedings of the IEEE International Conference on Computer Design*, page 34, 1986.

[9] F. Barahona, "On Via Minimization," *IEEE Transactions on Circuits and Systems*, 37(4), pp. 527–530, April 1990.

[10] M. Beardslee, C. Kring, R. Murgai, H. Savoj, R. K. Brayton, and A. R. Newton, "SLIP: A Software Environment for System Level Interactive Partitioning," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 280–283, 1966.

[11] J. Bhasker, and S. Sahni, "A Linear Algorithm to Find a Rectangular Dual of a Planar Graph," *Proceedings of 21st ACM/IEEE Design Automation Conference*, pp. 108–114, 1986.

[12] S. Bhingarde, A. Panyam, and N. Sherwani, "On Optimal Cell Models for Over-the-cell Routing," *to appear in the Proceedings of 6th International Conference on VLSI Design, Bombay, India*, January 1993.

[13] J. Bianks, "Partitioning by Probability Condensation," *Proceedings of Design Automation Conference*, pp. 758–761, 1989.

[14] H. Bollinger, "A Mature DA System for PC Layout," *Proceedings of first International Printed Circuit Conference*, 1979.

[15] K. S. Boothe, and G. S. Lueker, "Testing for Consecutive Ones Property, Interval Graphs and Graph Planarity using P Q-trees Algorithm," *Journal of Computer and System Science*, 13, pp. 335–379, 1979.

[16] D. G. Boyer, "Split Grid Compaction for Virtual Grid Symbolic Design System," *IEEE International Conference on Computer-Aided Design*, pp. 134–137, November 1987.

[17] D. G. Boyer, and N. Weste, "Virtual Grid Compaction Using the Most Recent Layers Algorithm," *IEEE International Conference on Computer-Aided Design*, pp. 92–93, September 1983.

[18] H. N. Brady, "An Approach to Topological Pin Assignment," *IEEE Transactions on Computer-Aided Design*, CAD-3, pp. 250–255, July 1984.

[19] D. Braun, J. Burns, S. Devadas, H. K. Ma, K. Mayaram, F. Romeo, and A. Sangiovanni-Vincentelli, "Chameleon: A New Multi-Layer Channel Router," *Proceedings of 23rd Design Automation Conference*, IEEE-86, pp. 495–502, 1986.

[20] M. A. Breuer, "A Class of Min-Cut Placement Algorithms," *Proceedings Design Automation Conference*, pp. 284–290, 1977.

[21] M. A. Breuer, "Min-Cut Placement," *J. Design Automation and Fault-Tolerant Computing*, pp. 343–382, October 1977.

[22] S. Brown, J. Rose, and Z. G. Vransic, "A Detailed Router for Field-Programmable Gate," *IEEE Transactions on Computer-Aided Design*, 11, pp. 620–628, May 1992.

[23] S. Burman, H. Chen, and N. Sherwani, "Improved Global Routing using λ-Geometry," *The Proceedings of 29th Annual Allerton Conference on Communications, Computing, and Controls*, October 1991.

[24] S. Burman, C. Kamalanathan, and N. Sherwani, "New Channel Segmentation Model and Routing Algorithm for High Performance FPGAs.," *to appear in the Proceedings of International Conference on Computer-Aided Design*, 1992.

[25] J. Burns, and A. R. Newton, "SPARCS: A New Constraint-Based IC Symbolic Layout Spacer," *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 534–539, May 1986.

[26] J. L. Burns, and A. R. Newton, "Efficient Constraint Generation for Hierarchical Compaction," *IEEE International Conference on Computer Design*, pp. 197–200, 1987.

[27] M. Burstein, and S. J. Hong, "Hierarchical VLSI Layout: Simultaneous Placement and Wiring of Gate Arrays," *Proceedings of VLSI*, 1983.

[28] M. Burstein, and R. Pelavin, "Hierarchical Channel Router," *Proceedings of 20th ACM/IEEE Design Automation Conference*, pp. 519–597, 1983.

[29] M. Buschbom, "MCM Thermal Challenges," *Surface Mount Technology*, pp. 30–34, 1990.

[30] R. Camposano, and R.K. Brayton, "Partitioning Before Logic Synthesis," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 324–326, 1987.

[31] H. Chan, P. Mazumder, and K. Shahookar, "Macro-cell and Module Placement by Genetic Adaptive Search with Bitmap-represented Chromosome," *Integration: the VLSI Journal*, 12(1), pp. 49–77, November 1991.

[32] H. M. Chan, and P. Mazumder, "A Genetic Algorithm for Macro Cell Placement," Technical report, Department of Electrical Engineering and Computer Science, University of Michigan, 1989.

[33] K. C. Chang, and D. H. Du, "Efficient Algorithms for Layer Assignment Problem," *IEEE Transactions on Computer-Aided Design*, CAD-6(1), pp. 67–78, January 1987.

[34] W. H. Chang, "Analytical IC metal-line capacitance formulas," *IEEE Transactions on Microwave Theory and Technology*, MTT-24, pp. 608–611, 1976.

[35] H. R. Charney, and D. L. Plato, "Efficient Partitioning of Components.," *Proceedings of the 5th Annual Design Automation Workshop*, pp. 16.0–16.21, 1968.

[36] G. Chartrand, and L. Lesniak, *Graphs and Digraphs*, Wadsworth and Brooks/Cole Inc., Monterey, 1986.

[37] A. Chatterjee, and R. Hartley, "A New Simultaneous Circuit Partitioning and Chip Placement Approach Based on Simulated Annealing," *Proceedings of Design Automation Conference*, pp. 36–39, 1990.

[38] H. H. Chen, "Trigger: A Three Layer Gridless Channel Router," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 196–199, 1986.

[39] H. H. Chen, and E. Kuh, "Glitter: A Gridless Variable-Width Channel Router," *IEEE Transactions on Computer-Aided Design.*, CAD-5(4), pp. 459–465, 1986.

[40] R. W. Chen, Y. Kajitani, and S. P. Chan, "A Graph-Theoretic Via Minimization Algorithm for Two-Layer Printed Circuit Boards," *IEEE Transactions on Circuits and Systems*, CAS-30(5), pp. 284–299, May 1983.

[41] Y. Chen, and M. Liu, "Three-Layer Channel Routing," *IEEE Transactions on Computer-Aided Design*, CAD-3(2), pp. 156–163, April 1984.

[42] Y. H. Chen, and David P. Lapotin, "Congestion Analysis for Wirability Improvement," *Research Report*, IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY, 1989.

[43] C. Cheng, and E. Kuh, "Module Placement Based on Resistive Network Optimization," *IEEE Transaction on Computer-Aided Design*, CAD-3, pp. 218–225, July 1984.

[44] C. Chiang, M. Sarrafzadeh, and C. K. Wong, "A Powerful Global Router: Based on Steiner Min-Max Trees," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 2–5, November 7-10 1989.

[45] C. Chiang, M. Sarrafzadeh, and C. K. Wong, "A Weighted-Steiner-Tree-Based Global Router," *Manuscript*, 1992.

[46] J. D. Cho, and M. Sarrafzadeh, "The Pin Redistribution Problem in Multichip Modules," *In the Proceedings of Fourth Annual IEEE International ASIC Conference and Exhibit* , pp. 9-2.1–9-2.4, September 1991.

[47] M. J. Ciesielski, and E. Kinnen, "An Optimum Layer Assignment for Routing in IC's and PCB's," *Proceedings of 18th Design Automation Conference*, pp. 733–737, June 1981.

[48] J. P. Cohoon, and P. L. Heck, "BEAVER: A Computational Geometry Based Tool for Switchbased Routing," *IEEE Transactions on Computer-Aided Design*, 7, pp. 684–697, June 1988.

[49] J. P. Cohoon, and W. Paris, "Genetic Placement," *Proc. IEEE International Conference On Computer-Aided Design*, pp. 422–425, 1986.

[50] R. Cole, and A. Siegel, "River routing every which way, but loose," *Proceedings of 25th Annual Symposium on Foundation of Computer Science*, pp. 65–73, 1984.

[51] J. Cong, "Pin Assignment with Global Routing," *Proceedings of International Conference on Computer-Aided Design*, pp. 302–305, 1989.

[52] J. Cong, M. Hossain, and N. Sherwani, "A Provably Good Multilayer Topological Planar Routing Algorithm In IC Layout Design," *to appear in the IEEE Transactions on Computer-Aided Design.*

[53] J. Cong, and C. L. Liu, "Over-the-Cell Channel Routing," *Proceedings of International Conference on Computer-Aided Design*, pp. 80–83, 1988.

[54] J. Cong, and C. L. Liu, "On the Over-the-cell Channel Routing Problem," *IEEE Transactions on Computer-Aided Design*, pp. 408–418, 1990.

[55] J. Cong, and C. L. Liu, "On the k-layer Planar Subset Problem and Topological Via Minimization Problem," *IEEE Transactions on Computer-Aided Design*, 10(8), pp. 972–981, 1991.

[56] J. Cong, and B. Preas, "A New Algorithm for Standard Cell Global Routing," *IEEE International Conference on Computer-Aided Design*, pp. 176–179, 1988.

[57] J. Cong, B. T. Preas, and C. L. Liu, "General Models and Algorithms for Over-the-Cell Routing in Standard Cell Design," *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pp. 709–715, June 1990.

[58] J. Cong, D. F. Wong, and C. L. Liu, "A New Approach to the Three-Layer Channel Routing Problem," *IEEE International Conference on Computer-Aided Design*, pp. 378–381, 1987.

[59] T. Cormen, C. E. Leiserson, and R. Rivest, *Introduction to Algorithms*, McGraw Hill, 1990.

[60] J. Cullum, W. Donath, and P. Wolfe, "The Minimization of Certain Non-differentiable Sums of Eigenvalues of Symmetric Matrices," *Mathematical Programming Study*, 3, pp. 35–55, 1975.

[61] W. Dai, and E. S. Kuh, "Simultaneous Floor Planning and Global Routing for Hierarchical Building-Block Layout," *IEEE Transactions on Computer-Aided Design*, pp. 828–837, September 1987.

[62] W. W. Dai, T. Dayan., and David Staepelaere, "Topological Routing in SURF: Generating a Rubber-Band Sketch," *Proceedings for the 28th Design Automation Conference* , pp. 39–44, 1991.

[63] W. W. Dai, B. Eschermann, E. Kuh, and M. Pedram, "Hierarchical Placement and Floorplanning in BEAR," *IEEE Transaction on Computer-Aided Design*, 8, pp. 1335–1349, Dec 1989.

[64] W. W. Dai, R. Kong, and J. Jue, "Rubber Band Routing and Dynamic Data Representation," *Proceedings for 1990 International Conference on Computer Aided Design* , pp. 52–55, 1990.

[65] W. W. Dai, R. Kong, and M. Sato, "Routability of a Rubber-Band Sketch," *Proceedings for the 28th Design Automation Conference* , pp. 45–48, 1991.

[66] W. W. Dai, and E. S. Kuh, *Global Spacing of Building-Block Layout, in C. H. Sequin, editor,* VLSI '87, Elsevier Science Publisher B. V., Amsterdam, The Netherlands, 1987.

[67] E. E. Davidson, "An Electrical Design Methodology for Multichip Modules," *In Proceedings for the International Conference on Computer Design*, pp. 573–578, 1984.

[68] V. K. De, *A Heuristic Global Router for Polycell Layout*, PhD thesis, Duke University, 1986.

[69] W. A. Dees, and P. G. Karger, "Automated Rip-up and Reroute Techniques," *Proceedings of Design Automation Conference*, 1982.

[70] D. N. Deutsch, "A Dogleg Channel Router," *Proceedings of 13th ACM/IEEE Design Automation Conference*, pp. 425–433, 1976.

[71] D. N. Deutsch, "Compacted Channel Routing," *Proceedings of IEEE International Conference on Computer-Aided Design*, International Conference on Computer-Aided Design-85, pp. 223–225, 1985.

[72] S. Dhar, M. A. Franklin, and D. F. Wang, "Reduction of Clock Delays in VLSI Structres," *Proceedings of IEEE International Conference on Computer Design*, pp. 778–783, October 1984.

[73] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, 1, pp. 269–271, 1959.

[74] H. N. Djidjev, "On the Problem of Partitioning Planar Graphs," *SIAM Jorunal on Algebraic and Discrete Methods*, 3(2), pp. 229–240, 1982.

[75] D. Dolev, K. Karplus, A. Siege, A. Strong, and J. D. Ullman, "Optimal Wiring Between Rectangle," *13th Annual ACM Symposium on Theory of Computation*, pp. 312–317, May 1987.

[76] W. E. Donath, R. J. Norman, B. K. Agrawal, S.E. Bello Sang Yong Han, J. M. Kurtzberg, P. Lowy, and R. I. McMillan, "Timing Driven Placement Using Complete Path Delays," *Proceedings of 27th ACM/IEEE Design Automation Conference*, pp. 84–89, 1990.

[77] D. H. C. Du, Oscar H. Ibarra, and J. Fernando Naveda, "Single-Row Routing with Crossover Bound," *IEEE Transactions on Computer-Aided Design*, CAD-6, pp. 190–201, 1987.

[78] D. H. C. Du, and L. H. Liu, "Heuristic Algorithms for Single Row Routing," *IEEE Transactions on Computers*, C-36, pp. 312–320, March 1987.

[79] A. E. Dunlop, and B. W. Kerninghan, "A Procedure for Placement of Standard-Cell VLSI Circuits," *IEEE Transactions on Computer-Aided Design*, pp. 92–98, January 1985.

[80] G. H. Ehrlich, S. Even, and R. E. Tarjan, "Intersection Graphs of Curves in the Plane," *Journal of Combinatorial Theory Series*, 21, pp. 394–398, April 1989.

[81] P. A. Eichenberger, *Fast Symbolic Layout Translation for Custom VLSI Integrated Circuits*, PhD thesis, Stanford University, Stanford, CA, 1986.

[82] R. J. Enbody, and H. C. Du, "Near-Optimal n-Layer Channel Routing," *Proceedings of the 23rd Design Automation Conference*, pp. 708–714, June 1986.

[83] S. Ercolani, and G. D. Micheli, "Technology Mapping for Electrically Programmable Gate Arrays," *Proceeding of 28th Design Automation Workshop*, pp. 234–239, 1991.

[84] B. Eschermann, "Hierarchical Placement for Macrocells with Simultaneous Routing Area Allocation," Technical Report Mem. UCB/ERL M88/49, Univ. Calif., Berkeley, 1988.

[85] A. E. Dunlop et. al., "Chip Layout Optimization Using Crititcal Path Weighting," *Proceedings of 21st ACM/IEEE Design Automation Conference*, pp. 133–136, 1984.

[86] A. El Gamal et. al, "An Architecture for Electrically Configurable Gate Arrays," *IEEE JSSC*, 24(2), pp. 394–398, April 1989.

[87] H. Hseih et. al, "A 9000-Gate User-Programmable Gate Array," *Proceedings of 1988 CICC*, pp. 15.3.1–15.3.7, May 1988.

[88] J. F. McDonald et al., "The Trail of Wafer-Scale Integration," *IEEE Spectrum*, pp. 32–39, October 1984.

[89] S. C. Wong et. al, "A 5000-Gate CMOS EPLD with Multiple Logic and Interconnect Arrays," *Proceedings of 1989 CICC*, pp. 5.8.1 – 5.8.4, May 1989.

[90] W. E. Donath et. al., "Timing Driven Placement Using Complete Path Delays," *Proceedings of 27th ACM/IEEE Design Automation Conference*, pp. 84–89, 1990.

[91] Y. Ogawa et. al., "Efficient Placement Algorithms Optimizing Delay for High-Speed ECL Masterslice LSI's," *Proceedings of 23rd ACM/IEEE Design Automation Conference*, pp. 404–410, 1986.

[92] S. Even, *Graph Algorithms*, Computer Science Press, 1979.

[93] S. Even, A. Pnueli, and A. Lempel, "Permutation Graphs and Transitive Graphs," *Journal of the ACM*, 19, pp. 400–410, 1972.

[94] M. Feuer, "VLSI Design Automation: An Introduction," *Proceedings of the IEEE*, 71(1), pp. 1–9, January 1983.

[95] C. M. Fiduccia, and R. M. Mattheyses, "A Linear-Time Heuristics for Improving Network Partitions," *Proceedings of the 19th Design Automation Conference*, pp. 175–181, 1982.

[96] D. Filo, J. C. Yang, F. Mailhot, and G. D. Micheli, "Technology Mapping for a Two-Output RAM-based Field Programmable Gate Arrays," *European Design Automation Conference*, pp. 534–538, February 1991.

[97] A. L. Fisher, and H. T. Kung, "Synchronizing Large Systolic Arrays," *Proceedings of SPIE*, pp. 44–52, May 1982.

[98] R. Fletcher, and C.M. Reeves, "Function Minimization by Conjugate Gradients," *Computer Journal*, 7, pp. 149–154, 1964.

[99] L. R. Ford, and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

[100] C. Fowler, G. D. Hachtel, and L. Roybal, "New algorithms for hierarchical place and route of custom VLSI," *International Conference on Computer-Aided Design*, pp. 273–275, 1985.

[101] R. J. Francis, J. Rose, and K. Chung, "Chortle: A Technology Mapping Program for Lookup Table-Based Field Programmable Gate Arrays.," *27th ACM/IEEE Design Automation Conference*, pp. 613–619, 1990.

[102] J. Frankle, and R. M. Karpp, "Circuit Placement and Cost Bounds by Eigenvector Decomposition," *Proc. IEEE International Conference On Computer-Aided Design*, pp. 414–417, 1986.

[103] C. P. Gabor, W. L. Hsu, and K. J. Supowit, "Recognizing Circle Graphs in Polynomial Time," *Proceedings 26th IEEE Symposium on Foundation of Computer Science*, pp. 106–116.

[104] N. H. Gabow, "A Almost Linear Time Algorithm for Two-processor Scheduling," *Journal of the ACM*, 29(3), pp. 766–780, 1985.

[105] T. Gao, P. M. Vaidya, and C. L. Liu, "A New Performance Driven Placement Algorithm," *Proceedings of International Conference on Computer-Aided Design*, pp. 44–47, 1991.

[106] J. Garbers, B. Korte, H. J. Promel, E. Schwietzke, and A. Steger, "VLSI-Placement Based on Routing and Timing Information," *European Design Automation Conference*, pp. 317–321, 1990.

[107] M. R. Garey, and D. S. Johnson, "The Rectilinear Steiner Tree Problem is NP-Complete," *SIAM Journal Applied Mathematics*, 32, pp. 826–834., 1977.

[108] M. R. Garey, and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, *Freeman, San Francisco*, 1979.

[109] M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou, "Unpublished Results," Technical report.

[110] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some Simplified NP-Complete Graph Problems," *Theory of Computation* , pp. 237–267, 1976.

[111] F. Gavril, "Algorithms for a Minimum Coloring, Maximum Clique, Minimum Covering by Cliques, and Maximum Independent Set of a Chordal Graph," *SIAM Journal of Computation*, 1, pp. 180–187, 1972.

[112] F. Gavril, "Algorithms for a Maximum Clique and a Maximum Independent Set of Circle Graph," *Network*, 3, pp. 261–273, 1973.

[113] S. Gerez, and O. Herrmann, "Switchbox Routing by Stepwise Refinement," *IEEE Transactions on Computer-Aided Design*, 8, pp. 1350–1361, Dec 1989.

[114] J. D. Giacomo, *VLSI Handbook: Silicon, Gallium Arsenide, and Superconductor circuits*, McGraw Hill, 1989.

[115] P. C. Gilmore, and A. J. Hoffman, "A Characterization of Comparability Graphs and of Interval Graphs," *Canadian Journal of Mathematics*, 16, pp. 539–548, 1964.

[116] M. K. Goldberg, and M. Burstein, "Heuristic Improvement Technique for Bisection of VLSI Networks," *IEEE International Conference on Computer Design*, pp. 122–125, 1983.

[117] M. C. Golumbic, "Complexity of Comparability Graph Recognition and Coloring," *Computing*, 18, pp. 199–208, 1977.

[118] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, *Academic Press*, 1980.

[119] T. F. Gonzalez, and S. Kurki-Gowdara, "Minimization of the Number of Layers for Single Row Routing with Fixed Street Capacity," *IEEE Transactions on Computer-Aided Design*, CAD-7, pp. 420–424, 1984.

[120] I. S. Gopal, D. Coppersmith, and C. K. Wong, "Optimal Wiring of Movable Terminals," *IEEE Transactions on Computers*, C-32, pp. 845–858, September 1983.

[121] S. Goto, "An Efficient Algorithm for the Two-Dimensional Placement Problem in Electrical Circuit Layout," *IEEE Trans. Circuits Syst.*, CAS-28, pp. 12–18, January 1981.

[122] J. Greene, V. Roychowdhury, S. Kaptanoglu, and A. E. Gamal, "Segmented Channel Routing," *27th ACM/IEEE Design Automation Conference*, pp. 567–572, 1990.

[123] J. Greene, and K. Supowit, "Simulated Annealing Without Rejected Moves.," *Proceedings of International Conference on Computer Design*, pp. 658–663, October 1984.

[124] H. J. Groeger, "A New Approach to Structural Partitioning of Computer Logic," *Proceedings of Design Automation Conference*, pp. 378–383, 1975.

[125] L. K. Grover, "Standard Cell Placement Using Simulated Sintering," *Proceedings of the 24th Design Automation Conference*, pp. 56–59, 1987.

[126] G. Gudmundsson, and S. Ntafos, "Channel Routing with Superterminals," *Proceedings of 25th Allerton Conference on Computing, Control and Communication*, pp. 375–376, 1987.

[127] U. I. Gupta, D. T. Lee, and J. Y. T. Leung, "Efficient Algorithms for Interval Graphs," *Networks*, 12, pp. 459–467, 1982.

[128] F. Hadlock, "Finding a Maximum Cut of a Planar Graph in Polynomial Time," *SIAM Journal of Computing*, 4, no. 3, pp. 221–225, September 1975.

[129] F. O. Hadlock, "A Shortest Path Algorithm for Grid Graphs," *Networks*, 1977.

[130] B. Hajek, "Cooling Schedules for Optimal Annealing," *Oper. Res.*, pp. 311–329, May 1988.

[131] K. M. Hall, "An r-Dimensional Quadratic Placement Algorithm," *Management Science*, 17, pp. 219–229, November 1970.

[132] G. T. Hamachi, and J. K. Ousterhout, "A Switchbox Router with Obstacle Avoidance," *Proceedings of 21st ACM/IEEE Design Automation Conference*, June 1984.

[133] S. E. Hambrusch, "Channel Routing Algorithms for Overlap Models," *IEEE Transactions on Computer-Aided Design*, CAD-4(1), pp. 23–30, January 1985.

[134] S. Han, and S. Sahni, "A Fast Algorithm for Single Row Routing," Technical Report 84-5, Department of Computer Science, University of Minnesota, Minneapolis, 1984.

[135] S. Han, and S. Sahni, "Single Row Routing in Narrow Streets," *IEEE Transactions on Computer-Aided Design*, CAD-3, pp. 235–241, July 1984.

[136] M. Hanan, "On Steiner's Problem With Rectilinear Distance," *SIAM Journal of Applied Mathematics*, 30(1), pp. 104–114, January 1976.

[137] M. Hanan, and J. M. Kurtzberg, "A Review of Placement and Quadratic Assignment Problems," *SIAM Rev.*, 14(2), pp. 324–342, April 1972.

[138] M. Hanan, A. Mennone, and P. K. Wolff, "An Interactive Man-Machine Approach to the Computer Logic Partitioning Problem," *Proceedings of Design Automation Conference*, pp. 70–81, 1974.

[139] M. Hanan, P. K. Wolff, and B. J. Agule, "Some Experimental Results on Placement Techniques," *J. Design Automation and Fault-Tolerant Computing*, 2, pp. 145–168, May 1978.

[140] S. Haruyama, and D. Fussell, "A New Area-Efficient Power Routing Algorithm for VLSI Layout," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 38–41, November 1987.

[141] A. Hashimoto, and J. Stevens, "Wire Routing by Optimization Channel Assignment Within Large Apertures," *Proceedings of the 8th Design Automation Workshop*, pp. 155–163, 1971.

[142] P. S. Hauge, R. Nair, and E. J. Yoffa, "Circuit Placement for Predictable Performance," *Proceedings of International Conference on Computer-Aided Design*, pp. 88–91, 1987.

[143] J. Heisterman, and T. Lengauer, "The Efficient Solution of Integer Programs for Hierarchical Global Routing," *IEEE Transactions on Computer-Aided Design*, CAD 10(6), pp. 748–753, June 1991.

[144] D. W. Hightower, "A Solution to the Line Routing Problem on a Continous Plane," *Proc. 6th Design Automation Workshop*, 1969.

[145] D. W. Hightower, "A Generalized Channel Router," *Proceedings of 17th ACM/IEEE Design Automation Conference*, pp. 12–21, 1980.

[146] D. Hill, and D. Shugard, "Global Routing Considerations in a Cell Synthesis System," *Proceedings of Design Automation Conference*, 1990.

[147] D. D. Hill, "ICON: A Toll for Design at Schematic, Virtual-grid and Layout Levels," *IEEE Design and Test*, 1(4), pp. 53–61, 1984.

[148] D. D. Hill, "A CAD System for the Design of Field Programmable Gate Arrays.," *28th ACM/IEEE Design Automation Conference*, pp. 187–192, 1991.

[149] R. B. Hitchcock, "Partitioning of Logic Graphs: A Theoretical Analysis of Pin Reduction," *Proceedings of Design Automation Conference*, pp. 54–63, 1970.

[150] J. Ho, M. Sarrafzadeh, G. Vijayan, and C. K. Wong, "Pad Minimization for Planar Routing of Multiple Power Nets," *IEEE Transactions on Computer-Aided Design*, CAD-9, pp. 419–426, 1990.

[151] J. M. Ho, M. Sarrafzadeh, G. Vijayan, and C. K. Wong, "Layer Assignment for Multichip Modules," *IEEE Transactions on Computer-Aided Design*, 9(12), pp. 1272–1277, December 1990.

[152] J. M. Ho, G. Vijayan, and C. K. Wong, "A New Approach to the Rectilinear Steiner Tree Problem," *IEEE Transactions on Computer-Aided Design*, 9(2), pp. 185–193, February 1985.

[153] J. M. Ho, G. Vijayan, and C. K. Wong, "Constructing the Optimal Rectilinear Steiner Tree Derivable from a Minimum Spanning Tree," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 5–8, November 1989.

[154] T. T. Ho, S. S. Iyengar, and S. Q. Zheng, "A General Greedy Channel Routing Algorithm," *IEEE Transactions on Computer-Aided Design*, 10(2), pp. 204–211, February 1991.

[155] N. Holmes, N. Sherwani, and M. Sarrafzadeh, "Algorithms for Over-the-Cell Channel Routing Using the Three Metal Layer Process," *IEEE International Conference on Computer-Aided Design*, 1991.

[156] N. Holmes, N. A. Sherwani, and M. Sarrafzadeh, "New Algorithm for Over-the-Cell Channel Routing Using Vacant Terminals," *Proceedings of 27th ACM/IEEE Design Automation Conference*, pp. 126–131, June 1991.

[157] M. Hossain, and N. A. Sherwani, "On Topological Via Minimization and Routing," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 532–534, November 10-14 1991.

[158] T. M. Hsieh, H. W. Leong, and C. L. Liu, "Two-Dimensional Layout Compaction by Simulated Annealing," *IEEE International Symposium on Circuits and Systems*, pp. 2439–2443, 1988.

[159] C. P. Hsu, "General River Routing Algorithm," *Proceedings of 20th Design Automation Conference*, pp. 578–583, June 1983.

[160] C. P. Hsu, "Minimum-Via Topological routing," *IEEE Transactions on Computer-Aided Design*, CAD-2(4), pp. 235–246, 1983.

[161] W. L. Hsu, "Maximum Weight Clique Algorithm for Circular-Arc Graphs and Circle Graphs," *SIAM Journal of Computation*, 14(1), pp. 160–175, February 1985.

[162] M. Y. Hsueh, "Symbolic Layout and Compaction of Integrated Circuits," Technical Report UCB/ERL M79/80, Electronics Research Laboratory, University of California, Berkeley, CA, 1979.

[163] M.Y. Hsueh, and D. O. Pederson, *Computer-Aided Layout of LSI Circuit Building-Blocks*, PhD thesis, University of California at Berkeley., December 1979.

[164] T. C. Hu, and M. T. Shing, *A Decomposition Algorithm for Circuit Routing in VLSI*, 1985.

[165] M. D. Huang, F. Romeo, and A. Sangiovanni-Vincentelli, "An Efficient General Cooling Schedule for Simulated Annealing," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 381–384, 1986.

[166] F. K. Hwang, "An $O(n \log n)$ Algorithm for Rectilinear Steiner Trees," *Journal of the Association for Computing Machinery*, 26(1), pp. 177–182, April 1976.

[167] F. K. Hwang, "On Steiner Minimal Trees With Rectilinear Distance," *SIAM Journal of Applied Mathematics*, 30(1), pp. 104–114, January 1976.

[168] F. K. Hwang, "An $O(n \log n)$ Algorithm for Suboptimal Rectilinear Steiner Trees," *Transactions on Circuits and Systems*, 26(1), pp. 75–77, January 1979.

[169] L. T. Hwang, and I. Turlik, "The Skin Effect in Thin-Film Interconnections for ULSI/VLSI Packages," *Technical Report Series TR91-13* , MCNC Research Triangle Park, NC 27709, 1991.

[170] M. A. B. Jackson, and E. S. Kuh, "Performance-Driven Placement of Cell Based IC's," *Proceedings of 26th ACM/IEEE Design Automation Conference*, pp. 370–375, 1989.

[171] M. A. B. Jackson, A. Sirinivasan, and E.S. Kuh, "Clock Routing for High-Performance ICs," *Proceedings of 27th ACM/IEEE Design Automation Conference*, pp. 573–579, June 1990.

[172] D. W. Jepsen, and C. D. Gelatt Jr., "Macro Placement by Monte Carlo Annealing," *Proceedings of IEEE International Conference on Computer Design*, pp. 495–498, 1983.

[173] S. C. Johnson, "Hierarchical Clustering Schemes," *Psychometrika*, pp. 241–254, 1967.

[174] R. Joobbani, *An Artificial Intelligence Approach to VLSI Routing*, Kluwer Academic Publisher, 1986.

[175] A. Joseph, and R. Y. Pinter, "Feed-Through River Routing," *Integration, the VLSI Journal*, 8, pp. 41–50, 1989.

[176] E. G. Coffman Jr., and R. L. Graham, "Optimal Scheduling for Two Processor Systems," *Acta Informatica*, 1, pp. 200–213, 1972.

[177] Kozminski. K, and E. Kinnen, "Rectangular Dual of a Planar Graph for Use in Area Planning for VLSI Integrated Circuits," *Proceedings of 21st ACM/IEEE Design Automation Conference*, pp. 655–656, 1984.

[178] A. Kahng, J. Cong, and G. Robins, "High-Performance Clock Routing Based on Recursive Geometric Matching," *Proceedings of 28th ACM/IEEE Design Automation Conference*, pp. 322–327, June 1991.

[179] Y. Kajitani, "On Via Hole Minimization of Routing in a 2-Layer Board," *Proceedings of IEEE international Conference on Circuits and Computers*, pp. 295–298, June 1980.

[180] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. V. Vazirani, and V. V. Vazirani, "Global Wire Routing in Two-Dimensional Arrays," *Algorithmica*, 1987.

[181] D. Karpenske, and C. Talbot, "Testing and Diagnosis of Multichip Modules," *Solid State Technology*, pp. 24–26, 1991.

[182] K. Karplus, "Amap: a Technology Mapper for Selector-based Field-Programmable Gate Arrays," *Proceedings of 28th ACM/IEEE Design Automation Conference*, pp. 244–247, 1991.

[183] K. Karplus, "Xmap: a Technology Mapper for Table-lookup Field-Programmable Gate Arrays.," *Proceedings of 28th ACM/IEEE Design Automation Conference*, pp. 240–243, 1991.

[184] T. Kawamoto, and Y. Kajitani, "The Minimum Width Routing of a 2-Row 2-Layer Polycell Layout," *Proceedings of 16th Design Automation Conference*, pp. 290–296, 1979.

[185] G. Kedem, and H. Watanbe, "Graph-Optimization Techniques for IC Layout and Compaction," *IEEE Transactions on Computer-Aided Design*, CAD-3(1), pp. 12–20, 1984.

[186] W. Kernighan, and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Technical Journal*, 49, pp. 291–307, 1970.

[187] W. A. Khan, M. Hossain, and N. A. Sherwani, "Zero Skew Routing in Multiple Clock Synchronous Systems," *to appear in the Proceedings of IEEE International Conference on Computer-Aided Design*, November 1992.

[188] K.-Y. Khoo, and J. Cong, "A Fast Multilayer General Area Router for MCM Designs," *To appear in* IEEE Transactions on Circuits and Systems.

[189] H. Kim, "Finding a Maximum Independent Set in a Permutation Graph," *Information Processing Letters*, 36, pp. 19–23, October 1990.

[190] S. Kirkpatrick, C. D. Gellat, and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, 220, pp. 671–680, May 1983.

[191] R. Kling, and P. Banerjee, "ESP: Placement by simulated evolution," *IEEE Trans. Computer-Aided Design*, 8(3), pp. 245–256, 1989.

[192] R. Kling, and P. Bannerjee, "ESP: A New Standard Cell Placement Package Using Simulated Evolution," *Proceedings of the 24th Design Automation Conference*, pp. 535–542, 1987.

[193] R. M. Kling, "Placement by Simulated Evolution," Ms thesis, Coordinated Science Lab.,College of Engr., Univ. of Illinois at Urbana-Champaign, 1987.

[194] N. L. Koren, "Pin Assignment in Auotmated Printed Circuit Board Design.," *proceedingd of the 9th Design Automation Workshop*, pp. 72–79, 1972.

[195] C. Kring, and A. R. Newton, "A Cell-Replicating Approach to Mincut-Based Circuit Partitioning.," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 2–5, November 1991.

[196] B. Krishnamurthy, "An Improved Mincut Algorithm for Partitioning VLSI Networks.," *IEEE Transactions on Computers*, pp. 438–446, 1984.

[197] J. B. Kruskal, "On the Shortest Spanning Subtree of A Graph and the Traveling Salesman Problem," *Proceedings of the American Mathematical Society*, 7(1), pp. 48–50, 1956.

[198] E. S. Kuh, T. Kashiwabara, and T. Fujisawa, "On Optimum Single Row Routing," *IEEE Transactions on Circuits Systems*, vol. CAS-26, pp. 361–368, June 1979.

[199] Y. S. Kuo, T. C. Chern, and W. Shih, "Fast Algorithm for Optimal Layer Assignment," *Proceedings of 25th ACM/IEEE Design Automation Conference*, pp. 554–559, June 1988.

[200] J. Lam, and J. Delosme, "Performance of a New Annealing Schedule," *Porceedings of the 25th Design Automation Conference*, 306-311 1988.

[201] D. P. LaPotin, "Early Assessment of Design, Packaging and Technology Tradeoffs," *International Journal of High Speed Electronics*, 2(4), pp. 209–233, 1991.

[202] E. L. Lawler, K. N. Levitt, and J. Turner, "Module Clustering to Minimize Delay in Digital Networks," *IEEE Transactions on Computers*, C-18(1), pp. 47–57, January 1969.

[203] A. Leblond, "CAF: A Computer-Assisted Floorplanning Tool," *Proceeding of 20th Design Automation Conference*, pp. 747–753, 1983.

[204] C. Y. Lee, "An Algorithm for Path Connections and Its Applications," *IRE Transactions on Electronic Computers*, 1961.

[205] D. T. Lee, J. M. Smith, and J. S. Liebman, "An $O(n \log n)$ Heuristic Algorithm Rectilinear Steiner Tree Problem," *Engineering Optimization*, Vol. 4(4), pp. 179–192, 1980.

[206] K. W. Lee, and C. Sechen, "A New Global Router for Row-Based Layout," *International Conference on Computer-Aided Design. IEEE.*, November 1988.

[207] C. E. Leiserson, and F. M. Maley, "Algorithms for routing and testing routability of planar VLSI layouts," *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pp. 69–78, 1985.

[208] C. E. Leiserson, and R. Y. Pinter, "Optimal Placement for River Routing," *SIAM Journal of Computing*, 12, No. 3, pp. 447–462, August 1983.

[209] T. Lengauer, "On the Solution of Inequality Systems Relevant to IC-Layout," *Journal of Algorithms*, 5, pp. 408–421, 1984.

[210] K. F. Liao, M. Sarrafzadeh, and C. K. Wong, "Single-Layer Global Routing," *In proceedings of the Fourth Annual IEEE International ASIC conference and Exhibit*, pp. 14–4.1–14–4.4, 1991.

[211] Y. Z. Liao, and C. K. Wong, "An Algorithm to Compact a VLSI Symbolic Layout with Mixed Constraints," *IEEE Transactions on Computer-Aided Design*, CAD-2(2), pp. 62–69, 1983.

[212] M.-S. Lin, H.-W. Perng, C.-Y. Hwang, and Y.-L. Lin, "Channel Density Reduction by Routing Over the Cells," *Proceedings of 28th ACM/IEEE Design Automation Conference*, pp. 120–125, June 1991.

[213] T. M. Lin, and C. A Mead, "Signal Delay in General RC Networks," *IEEE Transactions on Computer-Aided Design*, CAD-3, No. 4, pp. 331–349, October 1984.

[214] Y. L. Lin, Y. C. Hsu, and F. S. Tsai, "SILK: A Simulated Evolution Router," *IEEE Tranactions on Computer-Aided Design*, 8, pp. 1108–1114, October 1989.

[215] R. J. Lipton, and R. E. Tarjan, "A Separator Theorem for Planar Graphs," *SIAM Journal of Applied Mathematics*, 36(2), pp. 177–189, 1979.

[216] B. Lokanathan, and E. Kinnen, "Performance Optimized Floor Planning By Graph Planarization," *Proceedings of 26th ACM/IEEE Design Automation Conference*, pp. 116–121, 1989.

[217] R. D. Lou, M. Sarrafzadeh, and D. T. Lee, "An Optimal Algorithm for the Maximum Two-chain Problem," *Proceedings of* First SIAM-ACM Conference on Discrete Algorithms, 1990.

[218] W. K. Luk, "A Greedy Switchbox Router," *Integration, The VLSI Journal*, 3, pp. 129–149, 1985.

[219] C. Lursinsap, and D. Gajski, "An Optimal Power Routing for Top-Down Design Architecture," *Proceedings of the International Conference on Computer Design*, pp. 345–348, 1987.

[220] D. Maier, and J. A. Storer, "A Note on the Complexity of the Superstring Problem," Research Report No. 233, Computer Science Laboratory, Princeton University, 1977.

[221] F. M. Maley, *Single-Layer Wire Routing and Compaction*, The MIT Press, 1990.

[222] A. A. Malik, "An Efficient Algorithm for Generation of Constraint Graph for Compaction," *IEEE International Conference on Computer-Aided Design*, pp. 130–133, 1987.

[223] M. Marek-Sadowska, "An Unconstrained Topological Via Minimization Problem for Two-Layer Routing," *IEEE Transactions on Computer-Aided Design*, CAD-3(3), pp. 184–190, 1984.

[224] M. Marek-Sadowska, "Switch box routing: a retrospective," *INTEGRATION, The VLSI Journal*, 13, pp. 39–65, 1992.

[225] M. Marek-Sadowska, and S. P. Lin, "Timing Driven Placement," *Proceedings of International Conference on Computer-Aided Design*, pp. 94–97, 1989.

[226] M. Marek-Sadowska, and T. T. Trang, "Single-Layer Routing for VLSI: Analysis and Algorithms," *IEEE Transactions on Computer-Aided Design*, pp. 246–259, October 1983.

[227] J. M. Da Mata, "Allenda: A Procedural Language for the Hierarchical Specification of VLSI Layout," *Proceedings of the 22nd Design Automation Conference*, pp. 183–189, 1985.

[228] D. W. Matula, and F. Shahrokhi, "The Maximum Concurrent Flow Problem and Sparsest Cuts," *Tech. Report, Southern Methodist Univ.*, 1986.

[229] K. McCullen, J. Thorvaldson, D. Demaris, and P. Lampin, "A System for Floorplanning with Hierarchical Placement and Wiring," *European Design Automation Conference*, pp. 262–265, 1990.

[230] M. C. McFarland, "Computer-Aided Partitioning of Behavioral Hardware Description," *Proceedings of Design Automation Conference*, pp. 472–478, 1983.

[231] M. C. McFarland, "Using Bottom-Up Design Techniques in the Synthesis of Digital Hardware from Abstract Behavioral Descriptions," *Proc. of the 23rd Design Automation Conference*, pp. 474–480, 1986.

[232] C. Mead, and L. Conway, *Introduction to VLSI Systems, Chapter 1 MOS Devices and Circuits*, Addison Wesley, 1979.

[233] N. Metropolis, A. Rosenbluth, and M. Rosenbluth, "Equation of State Calculations by Fast Computing Machines," *Journal of chemistry and physics*, pp. 1087–1092, 1953.

[234] K. Mikami, and K. Tabuchi, "A Computer Program for Optimal Routing of Printed Circuit connectors," *IFIPS Proc.*, H47, pp. 1475–1478, 1968.

[235] G. L. Miller, "Finding Small Simple Cycle Separators for 2-Connected Planar Graph," *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pp. 376–382, 1984.

[236] L. L. Moresco, "Electronic System Packaging: The Search for Manufacturing the Optimum in a Sea of Constraints," *IEEE Transactions on Computers, Hybrids, and Manufacturing Technology*, pp. 494–508, 1990.

[237] L. Mory-Rauch, "Pin Assignment on a Printed Circuit Board," *Proceedings of the 15th Design Automation Conference*, pp. 70–73, 1978.

[238] A. S. Moulton, "Laying the Power and Ground Wires on a VLSI Chip," *Proceedings of the 20th Design Automation Conference*, pp. 754–755, 1983.

[239] A. Mukerjee, *Introduction to NMOS and CMOS VLSI Systems Design*, Prentice Hall, Englewood Cliffs, NJ, 1986.

[240] R. Murgai, R. K. Brayton, and A. Sangiovani-Vincentelli, "On Clustering for Minimum Delay/Area," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 6–9, November 1991.

[241] R. Murgai, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Improved Logic Synthesis Algorithms for Table Look Up Architectures.," *Proceedings of International Conference on Computer-Aided Design*, pp. 564–567, 1991.

[242] R. Murgai, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Performance Directed Synthesis for Table Look Up Programmable Gate Arrays.," *Proceedings of International Conference on Computer-Aided Design*, pp. 572–575, 1991.

[243] N. J. Naclerio, S. Masuda, and K. Nakajima, "Via Minimization for Gridless Layouts," *Proceedings of 24th Design Automation Conference*, pp. 159–165, June 1987.

[244] N. J. Naclerio, S. Masuda, and K. Nakajima, "Via Minimization Problem is NP-Complete," *IEEE Transactions on Computers*, 38(11), pp. 1604–1608, November 1989.

[245] R. Nair, "A Simple Yet Effective Technique for Global Wiring," *IEEE Transanctions on Computer-Aided Design*, CAD-6(2), 1987.

[246] S. Natarajan, N. Sherwani, N. Holmes, and M. Sarrafzadeh, "Over-the-cell Routing for High Performance Circuits," *Proceedings of 29th ACM/IEEE Design Automation Conference*, pp. 600–603, June 1992.

[247] T. Ohtsuki, *Partitioning, Assignment and Placement*, North-Holland, 1986.

[248] R. Okuda, T. Sato, H. Onodera, and K. Tamaru, "An Efficient Algorithm for Layout Compaction Problem with Symmetry Constraints," *IEEE International Conference on Computer-Aided Design*, pp. 148–151, 1989.

[249] J. Ousterhoust, "Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools," *IEEE Transactions on Computer-Aided Design,*, CAD-3, January 1984.

[250] C. H. Papadimitriou, and K. Steigliz, *Combinatorial Optimization - Algorithms and Complexity*, Prentice-Hall, Inc., 1982.

[251] A. M. Patel, "Partitioning for VLSI Placement Problems," *Proceedings of 18th ACM/IEEE Design Automation Conference*, pp. 137–144, 1981.

[252] M. Pedram, M. Marek-Sadowska, and E. S. Kuh, "Floorplanning with Pin Assignment," *Proceedings of International Conference on Computer-Aided Design*, pp. 98–101, 1990.

[253] R. Y. Pinter, "Optimal Layer Assignment for Interconnect," *1982 IEEE International Conference on Circuits and Computers*, pp. 398–401, September 1982.

[254] V. Pitchumani, and Q. Zhang, "A Mixed HVH-VHV Algorithm for Three-Layer Channel Routing," *IEEE Transactions on Computer-Aided Design*, CAD-6(4), 1987.

[255] A. Pnnueli, A. Lempel, and S. Even, "Transitive Orientation of Graphs and Identification of Permutation Graphs," *Canadian Journal of Mathematics*, 23, pp. 160–175, 1971.

[256] S. Poljak, "A note on Stable Sets and Coloring of Graphs," *Comment. Mathematics University Carolina*, 15, pp. 307–309, 1974.

[257] D.P. La Potin, and S. W. Director, "MASON: A Global Floorplanning Approach for VLSI Design," *IEEE Transaction on Computer-Aided Design*, pp. 477–489, October 1986.

[258] B. T. Preas, and M. J. Lorenzetti, *Physical Design Automation of VLSI Systems, Chap. 1, Introduction to Physical Design Automation,*, Benjamin Cummings, Menlo Park, CA, 1988.

[259] F. Preparata, and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[260] R. C. Prim, "Shortest Connection Networks and Some Generalizations," *Bell System Technical Journal*, 1957.

[261] N. R. Quinn, "The Placement Problem as Viewed from the Physics of Classical Mechanics," *Proceedings of the 12th Design Automation Conference*, pp. 173–178, 1975.

[262] N. R. Quinn, and M. A. Breuer, "A Force Directed Component Placement Procedure for Printed Circuit Boards," *IEEE Trans. Circuits and Syst.*, pp. 377–388, June 1979.

[263] R. Raghavan, and S. Sahni, "Single Row Routing," *IEEE Transactions on Computers*, C-32, pp. 209–220, March 1983.

[264] R. Raghavan, and S. Sahni, "Complexity of Single Row Routing Problems," *IEEE Transactions on Circuits and Systems*, CAS-31(5), pp. 462–471, May 1984.

[265] J. V. Rajan, *Automatic Synthesis of Microprocessors*, PhD thesis, Carnegie Mellon University, January 1989.

[266] J. V. Rajan, and D. E. Thomas, "Synthesis by Delayed Binding of Decisions," *Proc. of the 22nd Design Automation Conference*, pp. 367–373, 1985.

[267] P. Ramanathan, and K. G. Shin, "A Clock Distribution Scheme for Non-Symmetric VLSI Circuits," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 398–401, November 1989.

[268] J. Reed, A. Sangiovanni-Vincentelli, and M. Santamauro, "A New Symbolic Channel Router: YACR2," *IEEE Transactions on Computer-Aided Design*, CAD-4(3), pp. 208–219, 1985.

[269] M. L. Resnick, "SPARTA: A System Partitioning Aid," *IEEE Transactions on Computer-Aided Design*, pp. 490–498, 1986.

[270] D. Richards, "Complexity of Single-Layer Routing," *IEEE Transactions on Computers*, C-33(3), pp. 286–288, March 1984.

[271] C. S. Rim, T. Kashiwabara, and K. Nakajima, "Exact Algorithms for Multilayer Topological Via Minimization," *IEEE Transactions on Computer-Aided Design*, 8(4), pp. 1165–1184, November 1989.

[272] R. Rivest, and C. Fiduccia, "A Greedy Channel Router," *Proceedings of 19th ACM/IEEE Design Automation Conference*, pp. 418–424, 1982.

[273] F. Romeo, and A. Sangiovanni-Vincentelli, "Convergence and Finite Time Behavior of Simulated Annealing," *Proceedings of The 24th Conference on Decision and Control*, pp. 761–767, 1985.

[274] F. Romeo, A.S. Vincentelli, and C. Sechen, "Research on Simulated Annealing at Berekeley," *Proceedings of IEEE International Conference on Computer Design*, pp. 652–657, 1984.

[275] Jonathan Rose, "Parallel Global Routing for Standard Cells," *IEEE Transactions on Computer-Aided Design*, October 1990.

[276] H-J. Rothermel, and D. A. Mlynski, "Computation of Power Supply Nets in VLSI Layout," *Proceedings ACM/IEEE Design Automation Conference*, Proceedings of Design Automation Conference-81, pp. 37–47, 1981.

[277] F. Rubin, "The Lee Path Connection Algorithm," *IEEE Transactions on Computer-Aided Design*, CAD-3, No. 4, pp. 308–318, October 1974.

[278] Y. Saab, and V. Rao, "Stochastic Evolution: A Fast Effective Heuristic for Some Generic Layout Problems," *Proceedings of Design Automation Conference*, pp. 26–31, 1990.

[279] Y. G. Saab, and V. B. Rao, "An Evolution-Based Approach to Partitioning ASIC Systems," *Proceedings of Design Automation Conference*, pp. 767–770, 1989.

[280] M. G. Sage, "Future of Multichip Modules in Electronics," Proceedings of NEPCON West 89, 1989.

[281] S. Sahni, A. Bhatt, and R. Raghavan, "Complexity of Design Automation Problems," Technical Report 80-23, Department of Computer Science, University of Minnesota, MN, 1980.

[282] T. Sakurai, and T. Tamuru, "Simple formulas for two- and three- dimensional capacitances," *IEEE Transactions on Electron Devices*, ED-30, pp. 183–185, 1983.

[283] M. Sarrafzadeh, and D. T. Lee, "A New Approach to Topological Via Minimization," *IEEE Transactions on Computer-Aided Design*, 8, pp. 890–900, 1989.

[284] M. Sarrafzadeh, and R. D. Lou, "Maximum K-Coverings of Weighted Transitive Graphs with Applications," *IEEE International Symposium on Circuits and Systems*, pp. 332–335, 1990.

[285] M. Sarrafzadeh, and C. K. Wong, "Hierarchical Steiner Tree Construction in Uniform Orientations," Research report, Dept. of Electrical Engineering and Computer Science, Northwestern University, 1990.

[286] W. L. Schiele, "Improved Compaction by Minimized Length of Wires," *Proceedings of 20th ACM/IEEE Design Automation Conference*, pp. 121–127, 1983.

[287] W. L. Schiele, "Improved Compaction by Minimized Length of Wires," *Proceedings of Chapel Hill Conference on VLSI*, pp. 165–180, May 1985.

[288] M. Schlag, Y. Z. Liao, and C. K. Wong, "An Algorithm for Optimal Two Dimensional Compaction of VLSI Layouts," *Integration*, 1(2,3), pp. 179–209, September 1983.

[289] D. G. Schweikert, and B. Kernighan, "A Proper Model for the Partitioning of Electrical Circuits," *Proceedings of the 9th Design Automation Workshop*, pp. 57–62, 1972.

[290] W. Scott, and J. Outsterhout, "Plowing: Interactive Stretching and Compaction in Magic," *Proceedings of Design Automation Conference*, 1984.

[291] C. Sechen, and K. W. Lee, "An Improved Simulated Annealing Algorithm for Row-Based Placement," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 478–481, 1987.

[292] C. Sechen, and A. Sangiovanni-Vincentelli, "The Timber Wolf placement and routing package," *IEEE Journal of Solid-State Circuits*, Sc-20, pp. 510–522, 1985.

[293] K. Shahookar, and P. Mazumder, "A Genetic Approach to Standard Cell Placement," *First European Design Automation Conference*, March 1990.

[294] K. Shahookar, and P. Mazumder, "A Genetic Approach to Standard Cell Placement using Meta-Genetic Parameter Optimization," *IEEE Trans. Computer-Aided Design*, pp. 500–511, May 1990.

[295] N. Sherwani, and J. Deogun, "A New Heuristic for Single Row Routing Problems," *Proceedings of 26th ACM Design Automation Conference*, pp. 167–172, June 1989.

[296] N. Sherwani, and J. Deogun, "New Lower Bound for Single Row Routing Problems," *Proceedings of 1989 IEEE Midwest Symposium on Circuits and Systems*, August 1989.

[297] N. Sherwani, J. Deogun, and A. Roy, "A Parallel Algorithm for Single Row Routing Problems," *to appear in Journal of Circuits, Systems and Computers.*

[298] N. Sherwani, J. Deogun, and A. Roy, "Single Row Routing with Bounded number of Doglegs per Net," *Proceedings of 1989 IEEE International Symposium on Circuits and Systems*, pp. 43–46, May 1989.

[299] N. Sherwani, B. Wu, and M. Sarrafzadeh, "Algorithms for Minimum Bend Single Row Routing," *IEEE Transactions on Circuits and Systems*, 39(5), pp. 412–415, May 1992.

[300] N. A. Sherwani, and B. Wu, "Clock Layout of High Performance Circuits based on Weighted Center Algorithm," *Proceedings of Fourth IEEE International ASIC Conference and Exhibit*, pp. P15–5.1–5.4, September 1991.

[301] H. Shin, and A. Sangiovanni-Vincentelli, "MIGHTY: a Rip-up and Reroute Detailed 'Router," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 2–5, November 1986.

[302] H. Shin, A. L. Sangiovanni-Vincentelli, and C. H. Sequin, "Two-Dimensional Compaction By 'Zone Refining'," *Proceedings of 23rd Design Automation Conference*, pp. 115–122, June 1986.

[303] M. Shin, Ernest S. Kuh, and Ren-Song Tsay, "High-Performance-Driven System Partitioning on Multi-Chip Modules," *the proceedings of* 29^{th} *ACM/IEEE DAC*, pp. 53–56, June 1992.

[304] E. Shragowitz, and J. Keel, "A Global Router Based on Multicommodity Flow Model," *INTEGRATION: The VLSI Journal*, 1987.

[305] A. Siegel, and D. Dolev, "The Seperation for General Single-Layer Wiring Barriers," *Carnegie-Mellon Conference on VLSI Systems and Computations*, pp. 143–152, October 1981.

[306] H. C. So, "Some theoretical results on the routing of multilayer printed-wiring boards," *Proceedings of 1974 IEEE International Symposium on Circuits and Systems*, pp. 296–303, 1974.

[307] J. Soukup, "Fast Maze Router," *Proceedings of 15th Design Automation Conference*, pp. 100–102, 1978.

[308] M. Sriram, and S. M. Kang, "A New Layer Assignment Approach for MCMs," *Technical Report UIUC-BI-VLSI-92-01*, The Beckman Institute, University of Illinois at Urbana-Champaign, 1992.

[309] M. Stallmann, T. Hughes, and W. Liu, "Unconstrained Via Minimization for Topological Multilayer Routing," *IEEE Transactions on Computer-Aided Design*, CAD-1(1), pp. 970–980, September 1990.

[310] K. R. Stevens, and W. M. VanCleemput, "Global Via Elimination in Generalized Routing Environment," *Proceedings of International Symposium on Circuits and Systems*, pp. 689–692, 1979.

[311] A. J. Stone, "Logic Partitioning," *Proceedings of Design Automation Conference*, pp. 2–22, 1966.

[312] P. R. Suaris, and G. Kedem, "A Quadrisection Based Combined Place and Route Scheme for Standard Cells," *IEEE Transactions on Computer-Aided-Design*, March 1989.

[313] K. J. Supowit, "Finding a Maximum Planar Subset of a Set of Nets in a Channel," *IEEE Transactions on Computer-Aided Design*, CAD-6(1), pp. 93–94, January 1987.

[314] S. Sutanthavibul, E. Shragowitz, and J. Rosen, "An analytical approach to Floorplan Design and Optimization," *IEEE Transaction on Computer-Aided Design*, 10, pp. 761–769, June 1991.

[315] Z. Syed, and A. El Gamal, "Single Layer Routing of Power and Ground Networks in Integrated Circuits," *Journal of Digital Systems*, 6, pp. 53–63, 1982.

[316] A. A. Szepieniec, "Integrated placement/routing in sliced layouts," *Proceedings of 23rd ACM/IEEE Design Automation Conference*, pp. 300–307, 1986.

[317] T. G. Szymanski, "Dogleg Channel Routing is NP-Complete," *IEEE Transactions on Computer-Aided Design*, CAD-4, pp. 31–41, January 1985.

[318] R. Tarjan, *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, 1983.

[319] M. Terai, K. Takahashi, K. Nakajima, and K. Sato, "A New Model for Over-the-Cell Channel Routing with Three Layers," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 432–435, 1991.

[320] S. K. Tewksbury, *Wafer-Level System Integration: Implementation Issues*, Kluwer Academic Press, Boston, 1989.

[321] B. S. Ting, and E. S. Kuh, "An Approach to the Routing of Multilayer Printed Circuit Boards," *Proceedings of IEEE Symposium on Circuits and Systems*, pp. 902–911, 1978.

[322] B. S. Ting, E. S. Kuh, and I. Shirakawa, "The Multilayer Routing Problem: Algorithms and Necessary and Sufficient Conditions for the Single Row, Single Layer Case.," *IEEE Transactions on Circuits and Systems*, pp. 768–778, December 1976.

[323] M. Tompa, "An Optimal Solution to a Wire-Routing Problem," *Journal of Computer and System Sciences*, 23(2), pp. 127–150, October 1981.

[324] T. T. K. Trang, M. Marek-Sadowska, and E. S. Kuh, "An Efficient Single-Row Routing Algorithm," *IEEE Transactions on Computer-Aided Design*, pp. 178–183, July 1984.

[325] R. Tsay, "Exact Zero Skew," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 336–339, November 1991.

[326] R. Tsui, and R. Smith II, "A High Density Multilayer Printed Circuit Board Router Based on Necessary and Sufficient Conditions for Single Row Routing," *Proceedings of 18th ACM/IEEE Design Automation Conference*, pp. 372–381, June 1981.

[327] S. Tsukijama, E. S. Kuh, and I. Shirakawa, "An Algorithm for Single Row Routing with Prescribed Street Congestion," *IEEE Transactions on Circuits and Systems*, pp. 765–772, September 1982.

[328] T. Tuan, and S. L. Hakimi, "River Routing With a Small Number of Jogs," *SIAM Journal of Discrete Mathematics*, 3(4), pp. 585–597, November 1990.

[329] R. R. Tummala, "Electronic Packaging in the 1990's-A Perspective from America," *In IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, 14(2), pp. 262–271, June 1991.

[330] R. R. Tummala, and E. J. Rymaszewski, *Microelectronics Packaging Handbook*, Van Nostrand Reinhold, 1989.

[331] A. Vannelli, "An Adaptation of the Interior Point Method for Solving the Global Routing Problem," *IEEE Transanctions on Computer-Aided Design of Integrated Circuits*, CAD-10(2), 1991.

[332] M. P. Vecchi, and S. Kirkpatrick, "Global Wiring by Simulated Annealing," *IEEE Transanctions on Computer-Aided Design of Integrated Circuits*, CAD-2(4), 1983.

[333] G. Vijayan, H. H. Chen, and C. K. Wong, "On VHV-Routing in Channels with Irregular Boundaries," *IEEE Transactions on Computer-Aided Design*, CAD-8(2), 1989.

[334] G. Vijayan, and R. Tsay, "A New Method for Floorplanning Using Topological Constraint Reduction.," *IEEE Transactions on Computer-Aided Design*, pp. 1494–1501, December 1991.

[335] Y. Wei, and C. Cheng, "Towards Efficient Hierarchical Designs by Ratio Cut Partitioning," *International Conference on Computer-Aided Design*, pp. 298–301, 1989.

[336] S.-J. Well, J. Leroy, and R. Crappe, "An Efficient Two-Dimensional Compaction Algorithm for VLSI Symbolic Layout," *1990 European Design Automation Conference*, pp. 196–200, 1990.

[337] J. Williams, "STICKS - A Graphical Compiler for High Level LSI Design," *Proceedings of AFIPS*, pp. 289–295, 1978.

[338] N. Wirth, "What Can We Do about the Unnecessary Diversity of Notations for Synctactic Definitions?," *Communications of the ACM*, November 1977.

[339] D. F. Wong, and C. L. Liu, "A New Algorithm for Floorplan Design," *Proceedings of 23rd ACM/IEEE Design Automation Conference*, pp. 101–107, 1986.

[340] N. Woo, "A Heuristic Method for FPGA Technology Mapping Based on the Edge Visibility," *28th ACM/IEEE Design Automation Conference*, pp. 248–251, 1991.

[341] B. Wu, N. Holmes, N. Sherwani, and M. Sarrafzadeh, "Over-the-Cell Routers for New Cell Models," *Proceedings of 29th ACM/IEEE Design Automation Conference*, pp. 604–607, June 1992.

[342] B. Wu, and N. A. Sherwani, "Clock Routing for High-Performance Circuits using movable Clock Terminals," *Proceedings of Fourth International Conference on IC Design, Manufacture and Application*, pp. 94–100, September 1991.

[343] B. Wu, and N. A. Sherwani, "Effective Buffer Insertion of Clock Trees for High-Speed VLSI Circuits," *Microelectronics*, 23, pp. 291–300, July 1992.

[344] J. G. Xiong, "Algorithms for Global Routing," *Proceedings of Design Automation Conference*, 1986.

[345] X.-M. Xiong, and E. S. Kuh, "The Scan Line Approach To Power and Ground Routing," *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 6–10, November 1986.

[346] Y. Pan Y. C. Hsu, and W. J. Kubitz, "A Path Selection Global Router," *Proceedings of Design Automation Conference*, 1987.

[347] C. Yang, and D. F. Wong, "Optimal Channel Pin Assignment," *IEEE Transactions Computer-Aided Design*, CAD 10(11), pp. 1413–1423, November 1991.

[348] M. Yannakakis, and F. Gavril, "Edge Dominating Sets in Graphs - Unpublished," 1978.

[349] M. Yannakakis, and F. Gavril, "The Maximum k-Colorable Problem for Chordal Graphs," *Information Processing Letters*, pp. 133–137, January 1987.

[350] X. Yao, M. Yamada, and C. L. Liu, "A New Approach to the Pin Assignment Problem," *Proceedings of 25th ACM/IEEE Design Automation Conference*, pp. 566–572, 1988.

[351] D. C. Yeh, S. M. Kang, and V. B. Rao, "CMOS Logic Circuit Partitioning for Equal Chip complexity," *Proceedings of IEEE International Conference on Computer Design*, pp. 358–360, 1987.

[352] J. Yih, and P. Mazumder, "A Neural Network Design for Circuit Partitioning," *IEEE Transactions on Computer-Aided Design*, pp. 1265–1271, 1990.

[353] T. Yoshimura, and E. S. Kuh, "Efficient Algorithms for Channel Routing," *IEEE Transactions on Computer-Aided Design*, CAD-1(1), pp. 25–30, January 1982.

TAB 48

Manual of Style

The Essential Guide for Writers, Editors, and Publishers

14th Edition

The Chicago Manual of Style

Fourteenth Edition

The University of Chicago Press Chicago and London

He
e
all who
readers,
publish
Almc
this edi
change
gy. It is
ples, an
comput
prepara
design,
The
continu
this edi
First
Manua
more ir
usage, a
revisior
feature
tive pu
terms, (
tribes, a
foreign
and a r
for con
suffixes
The
Manua
was sca
logicall
15 now
tion, ar
preferr
Notes a
and ref
and th
range (
compu
data ba
tronic
(Contin

The University of Chicago Press, Chicago 60637
The University of Chicago Press, Ltd., London

First edition published 1906. Twelfth edition 1969
Thirteenth edition 1982. Fourteenth edition 1993
Printed in the United States of America
02 01 00 99 98 10 9 8 7 6 5

ISBN (cloth): 0-226-10389-7

Library of Congress Cataloging-in-Publication Data

University of Chicago Press.
The Chicago manual of style — 14th ed.
p. cm.
Includes bibliographical references and index.
1. Printing, Practical—United States—Style manuals.
2. Authorship—Handbooks, manuals, etc. 3. Publishers and publishing—United States—Handbooks, manuals, etc. I. Title.
Z253.U69 1993
808'.027'0973—dc20 92-37475
CIP

♾ The paper used in this publication meets the minimum requirements of the American National Standard for Information Sciences—Permanence of Paper for Printed Library Materials, ANSI Z39.48-1992.

6.4 The first part of the chapter is concerned with actual questions of spelling; the second part, with related questions about distinctive treatment of words and phrases, especially the use of italics and quotation marks. The chapter ends with a tabulation of some rules for spelling compound words (table 6.1).

SPELLING

Preferences of Special Groups

BRITISH VERSUS AMERICAN SPELLING

6.5 The practice of the University of Chicago Press is generally to change British spelling to American (e.g., *colour* to *color*) in books published under its imprint and composed in the United States. This is done because American compositors, American proofreaders, and American editors are far more likely to catch inconsistencies when they are departures from normal American spellings than when they are departures from less familiar British forms. Retaining British orthography is particularly perilous when heavy editing is called for.

SPELLINGS PECULIAR TO PARTICULAR DISCIPLINES

6.6 Although University of Chicago Press practice, as noted above, is to follow Webster's first-listed form for words with variant spellings, the variant may carry special connotations within certain disciplines, and these should be respected. For example, although *archaeology* is the first spelling given for the name of that science, and the one generally preferred, some specialists in North American studies insist on the spelling *archeology* when it is used in connection with their work. So, too, many bankers, as well as students of the banking and home-loan businesses, traditionally spell the word *installment* with one *l*—*instalment*—an acceptable variant that editors should feel no compulsion to change.

Plurals

GENERAL RULES

6.7 The plurals of most nouns are formed by the addition of *s* or *es*. When the noun ends in soft *ch* or in *s*, *sh*, *j*, *x*, or *z*, the plural inflection is *es*.

thumbs churches fixes ratios boys

Plurals of nouns ending in *y* preceded by a consonant are formed by replacing the *y* with *ies:*

babies navies specialties

The plurals of some nouns are formed irregularly:

women leaves cattle sheep

COMPOUND NOUNS

6.8 Closed, or solid, compound nouns—that is, compound nouns written as one word—form their plurals in the usual way. Solid compounds ending with the suffix *ful* generally take the inflection at the end of that suffix, although some dictionaries give as an alternative the inflection of the root word:

bagfuls, bagsful cupfuls, cupsful

6.9 Hyphenated and open compounds are regularly made plural by the addition of the plural inflection to the element that is subject to the change in number:

fathers-in-law	coups d'état
sergeants-at-arms	masters of art
courts-martial	
doctors of philosophy	*but* tam-o'-shanters

PLURALS IN DICTIONARIES

6.10 Standard dictionaries provide plural forms for listed nouns when those plurals are formed irregularly, including nouns with such troublesome endings as *o* and *ey*. Special cases usually not covered by dictionaries are discussed in the following paragraphs.

PROPER NOUNS

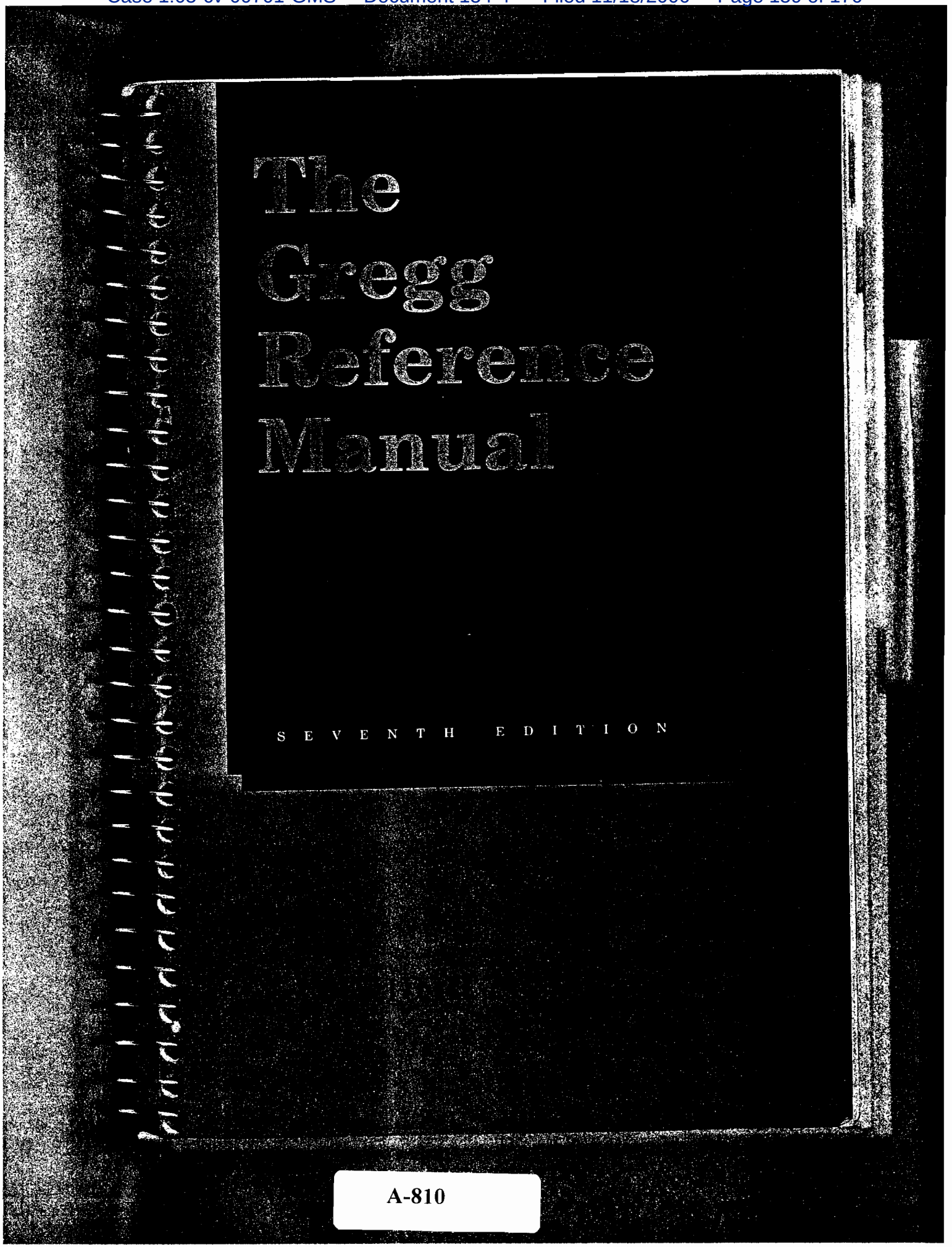
6.11 Names of persons and other proper nouns form the plural in the usual way, by adding *s* or *es:*

all the Edwards and Charleses
the Alexanders of modern times
flouting the Joneses
two Walden Ponds
rainy Sundays
three Marys

Note that the apostrophe is never used to denote the plural of a personal name: "The Schumachers left for London on Friday" (*not* "The Schumacher's . . .").

6.12 Exceptions to the general rule on adding *s* or *es* sometimes have to be made when the ending would suggest a false pronunciation. French

TAB 49


The
Gregg
Reference
Manual
SEVENTH EDITION

Library of Congress Cataloging-in-Publication Data

Sabin, William A.
The Gregg reference manual/William A. Sabin—7th ed.
p. cm.
Includes index.
ISBN 0-02-819920-0 (hardcover text ed.)—ISBN
0-02-819921-9 (softcover text ed.)—ISBN 0-02-819922-7
(spiral bound text ed.)
1. English language—Business English—Handbooks, manuals, etc.,
2. English language—Grammar—1950—Handbooks, manuals, etc.
3. English language—Transcription—Handbooks, manuals, etc.
4. Business writing—Handbooks, manuals, etc. I. Title.
PE1479.B87S23 1992
808'.042—dc20 92-2544
CIP

The Gregg Reference Manual, Seventh Edition

Imprint 1993

Send all inquiries to:
GLENCOE DIVISION
Macmillan/McGraw-Hill
936 Eastwind Drive
Westerville, Ohio 43081

ISBN 0-02-819921-9 (softcover text edition)
ISBN 0-02-819922-7 (spiral-bound text edition)

Printed in the United States of America.

5 6 7 8 9 0 A-HAW 99 98 97 96 95 94 93

SECTION 6

Plurals and Possessives

Forming Plurals

Basic Rule (¶601)
Nouns Ending in *S, X, CH, SH,* or *Z* (¶¶602–603)
Nouns Ending in *Y* (¶¶604–605)
Nouns Ending in *O* (¶¶606–607)
Nouns Ending in *F, FE,* or *FF* (¶608)
Nouns With Irregular Plurals (¶¶609–610)
Compound Nouns (¶¶611–613)
Foreign Nouns (¶614)
Proper Names (¶¶615–617)
Personal Titles (¶¶618–619)
Abbreviations, Letters, Numbers, and Words (¶¶620–626)

Forming Possessives

Possession Versus Description (¶¶627–629)
Singular Nouns (¶¶630–631)
Plural Nouns (¶¶632–633)
Compound Nouns (¶¶634–635)
Pronouns (¶¶636–637)
Abbreviations (¶638)
Personal and Organizational Names; Titles (¶¶639–640)
Nouns in Apposition (¶641)
Separate and Joint Possession (¶¶642–643)
Possessives Standing Alone (¶644)
Inanimate Possessives (¶¶645–646)
Possessives Preceding Verbal Nouns (¶647)
Possessives in *Of* Phrases (¶648)
Possessives Modifying Possessives (¶649)
Possessives in Holidays (¶650)
Miscellaneous Expressions (¶651)

¶601

Forming Plurals

When you are uncertain about the plural form of a word, consult a dictionary. If no plural is shown, form the plural according to the rules in ¶¶601–605.

Basic Rule

601 Plurals are regularly formed by adding *s* to the singular form.

suburb	suburbs	quota	quotas
fabric	fabrics	idea	ideas
yield	yields	committee	committees
egg	eggs	lie	lies
length	lengths	league	leagues
check	checks	alibi	alibis
rhythm	rhythms	menu	menus
flight	flights	bayou	bayous

NOTE: A few words have the same form in the plural as in the singular. (See ¶¶603, 1013, 1016, 1017.)

Nouns Ending in *S, X, CH, SH,* or *Z*

602 When the singular form ends in *s, x, ch, sh,* or *z,* the plural is formed by adding *es* to the singular.

bias	biases	tax	taxes
summons	summonses	sketch	sketches
business	businesses	wish	wishes
process	processes	quartz	quartzes

603 Singular nouns ending in silent *s* do not change their forms in the plural. (However, the *s* ending is pronounced when the plural form is used.)

one corps	two corps	a rendezvous	many rendezvous

Nouns Ending in *Y*

604 When a singular noun ends in *y* preceded by a *consonant,* the plural is formed by changing the *y* to *i* and adding *es* to the singular.

copy	copies	liability	liabilities
policy	policies	proxy	proxies

605 When a singular noun ends in *y* preceded by a *vowel,* the plural is formed by adding *s* to the singular.

delay	delays		guy	guys
attorney	attorneys	BUT:	soliloquy	soliloquies
boy	boys		colloquy	colloquies

Nouns Ending in *O*

606 Singular nouns ending in *o* preceded by a *vowel* form their plurals by adding *s* to the singular.

stereo	stereos	shampoo	shampoos
ratio	ratios	boo	boos
portfolio	portfolios	tattoo	tattoos
scenario	scenarios	duo	duos

TAB 50

The TimberWolf Placement and Routing Package

CARL SECHEN AND ALBERTO SANGIOVANNI-VINCENTELLI, FELLOW, IEEE

Abstract—TimberWolf is an integrated set of placement and routing optimization programs. The general combinatorial optimization technique known as simulated annealing is used by each program. Programs for standard cell, macro/custom cell, and gate-array placement, as well as standard cell global routing have been developed. Experimental results on industrial circuits show that area savings over existing layout programs ranging from 15 to 62 percent are possible.

I. Introduction

TIMBERWOLF is an integrated set of placement and routing optimization programs. Extensions and modifications of the general combinatorial optimization technique known as simulated annealing [1] are used by each program. Four basic optimization programs of the TimberWolf package have been developed.

1) A Standard-Cell Placement Program: This program places standard cells into rows and/or columns in addition to allowing user-specified macro blocks and pads. The program was interfaced to the CIPAR standard cell placement package developed by American Microsystems, Inc. For the largest circuits tested (800 to 2700 cells), TimberWolf reduced total estimated wire lengths by 45 to 66 percent in comparison with CIPAR alone. Furthermore, final chip areas were reduced by 30 to 57 percent as a result of the improved placement. For a circuit of 1000 cells, TimberWolf reduced the final chip area by 31 percent in comparison to CIPAR and by 21 percent over another commercially available standard cell placement program in a benchmark performed at AMI.

2) A Standard Cell Global Router Program: The global router reduced by 10 to 15 percent the number of wiring tracks used by the CIPAR router. This translated to an overall area savings for 6 to 8 percent. Vecchi and Kirkpatrick [2] recently described the use of simulated annealing for global routing.

3) A Macro/Custom Cell Placement Program: This program places cells of any rectilinear shape. Furthermore, the cells may have fixed geometry including pin locations (macro cells) or they may have fixed area with a given aspect ratio range and with pins that need to be placed (custom cells). All rotations and reflections of each cell are considered. TimberWolf also has the ability to place cells among user-defined subregions of the chip. TimberWolf allows multiple chips to be placed simultaneously. This package can also be used to place circuits on one or more printed circuit boards.

The macro/custom cell placement program is currently under test on industrial circuits. However, the program has been tested on a Honeywell Information Systems Italy printed circuit board. The processor board required the placement of 613 variable-sized circuits. TimberWolf reduced the total wire length by 10 percent over the manually placed board.

4) A Generalized Gate-Array Placement Program: This program allows user-specified macros and primary terminals. This program found placement with a 6- to 27-percent reduction in total estimated wire length for several benchmark problems in comparison to the best published results. This program optionally includes in the cost calculation a measure of the local routing congestion.

This paper presents the algorithms used by each of the programs comprising the TimberWolf package and also presents the results that have been obtained. In particular, Section II describes the basic algorithm. In Section III, the standard cell placement optimization algorithm and program are described. Section IV presents details on the standard cell global router. In Section V, the macro/custom placement optimization algorithm and program are described and in Section VI the gate-array placement algorithm and implementation are presented. Finally, Section VII is devoted to concluding remarks and future research.

II. The Basic Algorithm

Simulated annealing has been proposed by Kirkpatrick *et al.* [1] as an effective method for the determination of global minima of combinatorial optimization problems involving many degrees of freedom. Its basic feature is the possibility of exploring the configuration space of the optimization problem allowing *hill climbing* moves, i.e., the acceptance of new configurations of the problem which increase the cost. These moves are controlled by a parameter, in analogy with temperature in the annealing process, and are less and less likely towards the end of the process.

Manuscript received August 31, 1984; revised December 18, 1984. The algorithmic part of this research has been supported by DARPA under Grant N00039-83-C-0107. The TimberWolf placement and routing package has been supported by a Grant from the MICRO of the State of California.

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

0018-9200/85/0400-0510$01.00 ©1985 IEEE

Given a combinatorial optimization problem specified by a finite set of configurations or states S and by a cost function c defined on all the states j in S, the simulated annealing algorithm is characterized by a rule to generate randomly a new state or configuration with a certain probability, and by a random acceptance rule according to which the new configuration is accepted or rejected. A parameter T controls the acceptance rule.

The basic structure of the algorithm is presented in the next subsection. Theoretical investigations of the simulated annealing optimization technique have been reported by our research group [3] and elsewhere [4], [5].

A. Algorithm Structure

The following function gives the general structure of the class of algorithms called *probabilistic hill-climbing algorithms* of which simulated annealing is a special case. This class has been proposed in [3] where a number of different algorithms with the same structure have been introduced.

```
Algorithm Structure (j0, T0){
  /*
  * Given an initial state j0 and an
  * initial value for the parameter T,
  * T0,
  */
  T = T0;
  X = j0;
    while("stopping criterion" is not satisfied){
        while("inner loop criterion" is not satisfied){
            j = generate(X);
            /*
            * generate is a function which
            * returns a new state j generated
            * incrementally from the previous state
            * X by a weighted random selection.
            */
            if (accept(c(j), c(X), T){
            X = j;
            }
        }
        T = update(T);
    }
}
```

The acceptance of a new state j is determined by ***accept***, whose structure is shown below.

```
accept(c(j), c(i), T){
            /*
            * Returns 1 if the cost variation passes a test
            * T is the control parameter
            */
            Δc = c(j) − c(i);
            y = f(Δc, T);
            r = random(0,1);
                /*
                * random is a function which returns a
                * pseudo-random number uniformly
                * distributed on the interval [0,1]
                */
            if(r < y){
              return(1);
            }else{
              return(0);
            }
        }
}
```

The algorithms in the class described above are characterized by 1) the generation function **generate**, 2) the acceptance function **accept**, 3) the updating function **update**, 4) the inner loop criterion, and 5) the stopping criterion. In the original version of simulated annealing, the acceptance function is governed by the function f shown below

$$f(\Delta c, T) = \min[1, \exp(-\Delta c/T)].$$

It is possible to vary the shape of f by adjusting the control parameter T, called temperature. The updating rule for T is given below.

$$T_{\text{new}} = \alpha(T_{\text{old}}) * T_{\text{old}}, \qquad 0 < \alpha < 1.$$

In the function **accept**, note that new states characterized by $\Delta c \leq 0$ always satisfy the acceptance criterion. However, for the new states characterized by $\Delta c > 0$, the parameter T plays a fundamental role. If T is very large, then r is likely to be less than y and a new state is almost always accepted irrespective of Δc. If T is small, close to 0, then only new states that are characterized by very small $\Delta c > 0$ have any chance of being accepted. In general, all states with $\Delta c > 0$ have smaller chances of satisfying the test for smaller values of T.

The properties of this class of algorithms can be studied using Markov chains as the theoretical models. Theoretical analysis [3] shows that this class generates with probability 1.0 the global optimum of the optimization problem, provided that certain conditions on the number of iterations at each T or a certain updating rule for T is followed. These results are unfortunately asymptotic and provide little information on how to choose the various parameters for the implementation of the algorithm. However, they serve to give confidence in the well posedness of the algorithm and to provide some insight on the reasons why simulated annealing has performed well in practical cases. In the remainder of the paper, attention will be given to the actual implementation of the various functions, the inner loop criterion, and the stopping criterion.

The best results with simulated annealing have been obtained in our experiments by starting with a large value of the parameter T, whereby virtually all proposed new states are accepted. Further, the best results have been obtained when the system is allowed to achieve "equilibrium" at each stage (or value of T) of the annealing process. That is, a sufficient number of iterations are performed in the inner loop such that the probability

distribution of the configuration is "close" to the stationary probability distribution of the Markov chains associated with the algorithm (see [3] for more details on the theory of simulated annealing). This is implemented by the "inner loop criterion" in the simulated annealing algorithm. The "stopping criterion" is satisfied when the cost function's value remains the same after several stages of the annealing process.

In simulated annealing, the best results have been obtained when the parameter T is slowly reduced when the cost function's value begins to decrease significantly. For each successive step of the annealing process, T is lowered exponentially. The TimberWolf programs currently allow the value of α to be specified for each value of T. The value of α is usually in the range of 0.8–0.95.

B. The TimberWolf Implementation of the Simulated Annealing Algorithm

For the applications of interest here, little difference was noted when using different functions f in the acceptance function **accept**. Hence the standard form for f as proposed by [1] was used. This section presents the TimberWolf implementations of the other functions.

1) Generating New States: The TimberWolf programs begin with a random initial placement or wiring configuration. A new state is generated by either exchanging two fundamental units or moving a unit to another location. For the gate array placement program, the new state is generated by the interchange of two *modules*, where a module refers to a fundamental unit specified in the net list. The standard cell placement program also generates new states by the interchange of cells. However, because standard cells typically vary in width, the interchange of two cells often results in a non-feasible solution because overlaps are not allowed. This is solved by a penalty function approach, first described by Kirkpatrick, Gelatt, and Vecchi [1]. The TimberWolf implementation of this approach will be described in the next section. The penalty function approach is also employed by the macro/custom cell placement program because the cells typically vary in both height and width.

For the standard cell and macro/custom cell problems, new states are also generated by the movement of a cell to a new location. Experimental investigation has revealed that the use of both methods of generating new states is necessary to achieve the best results. Furthermore, orientation changes of standard and macro/custom cells are performed which result in new states. If allowed by the user, new states are also generated for custom cells by assigning a new location to a pin or group of pins and by changing the aspect ratio of the cell.

For the standard cell global router program, new states are generated by assigning a portion of a net to a different channel.

2) Cost Function: The cost function for the placement programs is based on total estimated wire length. The standard cell and macro/custom cell programs also include a penalty function term which penalizes overlaps of the cells. The cost function for the standard cell global router is based on the estimated wiring area which is approximated by the total channel density, that is, the sum over all channels of the channel density.

3) Generating New Values of T: In the current implementation of TimberWolf, the parameter α is user-specified as α versus T data. The best results have been obtained when α is the largest (approximately 0.95) during the stages of the algorithm when the cost function is decreasing rapidly. Furthermore, the value of α is given its lowest values at the initial and latter stages of the algorithm (usually 0.80). The value of α is gradually increased from its lowest value to its highest value, and then gradually decreased back to its lowest value.

4) The Inner Loop Criterion: The inner loop criterion is implemented by the specification of the number of new states generated for each stage of the annealing process. This number is specified as a multiple of the number of fundamental units for the placement or routing problem. For the gate array placement and standard cell global router programs, 20 new states per unit are generated at each stage. The standard cell and macro/custom cell placement problems have many more degrees of freedom (orientation changes, pin location changes, etc.) and hence 100 or more new states are generated per cell at each stage.

5) The Stopping Criterion: The stopping criterion is implemented by recording the cost function's value at the end of each stage of the annealing process. The stopping criterion is satisfied when the cost function's value has not changed for 3 consecutive stages.

III. Standard Cell Placement Optimization Program

A. Introduction

TimberWolf is applicable to standard cell placement problems of the complexity shown in Fig. 1. TimberWolf optimizes the placement of standard cells into row and/or column blocks. Furthermore, the various blocks may have differing heights. The program also optimizes the placement of pads or buffer circuitry, as well as macro blocks. The macro blocks may be positioned anywhere on the chip. The estimation of the wire length for a single net is determined by computing the half-perimeter of the bounding box of the net. The bounding box is defined by the smallest rectangle which encloses all of the pins comprising the net. For the case of a two-pin net, this is the Manhattan distance. Because exact pin locations are used in the wire length calculations, TimberWolf considers all possible orientations for a cell, pad, or macro block. A group of pins which are internally connected within a cell must be given to TimberWolf as a single pin with a location which is the average of the locations of its constituent pins.

The program employs the *exchange class mechanism* for blocks as well as cells, pads and macros. If two blocks have

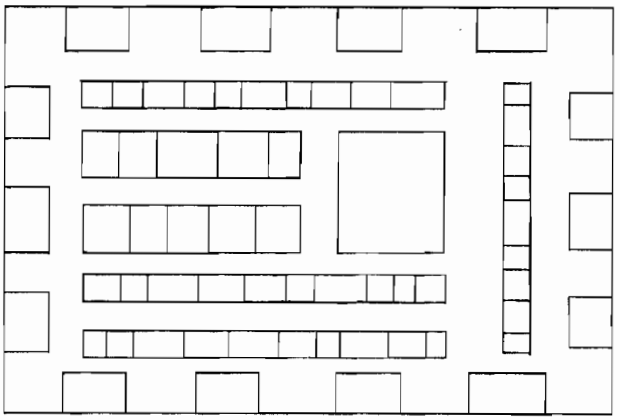


Fig. 1. Example of a general standard cell layout to which TimberWolf is applicable.

the same exchange class, then cells from these blocks are interchangeable. Blocks with differing exchange classes may not have their cells interchanged. Differing exchange classes for blocks are usually employed when blocks have different heights. Furthermore, two cells or two pads may be interchanged only if they belong to the same exchange class.

An additional feature is the net-weighting capability. For any given net, it is possible to weight the *horizontal span* of the net separately from the *vertical span* of the net. The horizontal span of a net is defined as the span of the smallest rectangle which encloses all of the pins comprising the net (bounding box) in the x direction of the x-y coordinate system. Similarly, the vertical span of a net is defined as the span of the bounding box in the y direction of the x-y coordinate system. For critical nets, it is usual to increase both the horizontal weight and the vertical weight, hence ensuring that these nets are kept as short as possible. For double-metal circuits, it is often the case that there are many uncommitted route throughs present in each cell. Consequently, vertical net spans are in some sense *cheaper* than horizontal net spans (which require the allocation of horizontal channel tracks and their associated area). In this case, the best results have been obtained when the vertical weights for the nets are made smaller in comparison to the horizontal weights.

B. Algorithm Details

The cost function for the simulated annealing algorithm consists of two independent portions. The first portion is the total estimated wire length. The second portion is the penalty function which consists of a total sum of overlap penalties. This penalty function was incorporated because of the usual difference in width of the standard cells. Often two cells are selected for interchange which differ in width. Therefore, an exchange of location of these two cells often results in some overlap with one or more of the other cells. Furthermore, the program often selects a single cell for a displacement to a new location. Once again, some overlap may result. The exchange of cells or the displacement of a single cell may also result in a portion of a cell dangling off the end of a row or column block. This is treated as a case of overlap with an *imaginary cell* being located at the ends of each column and row block. This feature increases the number of states in the state space S. Experimental investigation has shown that this results in better placements.

When two standard cells overlap, a penalty is assessed which is proportional to the square of the quantity of the amount of linear overlap plus an offset parameter. The offset parameter is chosen to ensure that when the parameter T approaches zero, then the total amount of overlap approaches zero. A larger value of the offset parameter generally results in more uniform block lengths at the expense of increased total wire length. On the other hand, a smaller value generally results in the smaller values of total wire length with less uniformity of block lengths. Experimentally it has been observed that setting the offset value to 3 yields the best overall results.

The overlap penalty function has an additional term which controls block lengths. The sum of the lengths of the cells in a particular block is compared to the actual block length. A penalty is assessed which is equal to the absolute value of the difference times a parameter value. As an example, consider the movement of a cell from a block whose total cell length is greater than the actual length of the block to another block whose total cell length is less than its actual length. The penalty term is reduced in this case. On the other hand, moving a cell from a block whose total cell length is less than its actual length to a block whose total cell length is greater than its actual length increases the penalty term. It has been experimentally observed that a parameter value of 5 results in very uniform block lengths with no compromise in the final total wire length.

The alternative to the aforementioned overlap concept is of course to not allow overlaps. For example, when inserting a cell into a row block, if insufficient space is available then the cells to the right are all shifted farther to the right as necessary. This has the obvious disadvantage of destroying the relationships between the shifted cells and the cells on the neighboring rows. The overlap concept was employed so as to not disturb the placement of the remaining cells when performing an interchange of cells or a displacement of a single cell.

The selection of new states is based on the following considerations: 1) A random number between one and the total number of cells, pads and macro blocks is generated. The cells are numbered from one to the total number of cells, and the pads and macro blocks are numbered starting from the number of cells plus one. If the random number is less than or equal to the number of cells, then a cell is selected. Otherwise, a pad or macro block is selected. 2) A second random number is selected between 1 and the total number of cells, pads, and macro blocks. 3) If the two numbers selected both represent cells, then the pair of cells are interchanged to generate a new state. 4) Similarly, if two pads or two macro blocks were selected, then an interchange constitutes the new state. 5) If the two numbers selected do not represent the same unit (that is, cell,

pad, or macro block) then the first unit selected governs the generation of a new state. If this first unit was a pad or macro block, then an orientation change of the respective unit is attempted. If the first unit was a cell, then this cell is displaced to a new location. If this new state is rejected, then the next state generated is an orientation change for the cell.

The ratio of single cell displacements to cell interchanges has a pronounced effect on the quality of the final placement. Experimental investigation has revealed that a ratio of about 5 to 1 yields the best results. Hence, if the first unit selected was a cell, the generation of the second random number is weighted to produce the desired ratio. This is implemented by generating a random number between one and the number of cells multiplied by 5.

The displacement of a cell to a new location is controlled by a *range limiter*, which limits the range of the displacement of a cell. For example, in the latter stages of the algorithm when the value of T approaches zero, the displacement of a cell has very little chance of being accepted unless the displacement is very local. By limiting the range of the cell displacements in the latter stages of the algorithm, the cells undergo many small displacements while gradually eliminating overlaps and reducing wire length.

The implementation of the range limiter is as follows. A rectangular *window* is centered at the center of the cell to be displaced and this window has a particular horizontal span and a particular vertical span. At the beginning of the algorithm, when T is at its maximum value, the horizontal span of the window is equal to twice the horizontal span of the chip and similarly the vertical span of the window is equal to twice the vertical span of the chip. The horizontal and vertical window spans are proportional to the logarithm of the value of T. Hence, when the value of T is reduced, the size of the window is correspondingly reduced. When a cell is to be displaced, a randomly-selected location within the window is chosen as the new location for the cell. That is, a block (row or column) is randomly selected which intersects the window and then a random position is selected within that block and within the window.

Pairwise interchanges of cells are also controlled by the range limiter. An interchange of two cells is attempted only if the window can be positioned such that it contains the centers of both cells.

As T is reduced, eventually the size of the range-limiter window has been reduced such that inter-block cell displacements or interchanges are no longer attempted. At this point, all residual cell overlaps are removed and the blocks are compacted. The generation of new states then takes on a different form as follows: 1) A standard cell is randomly selected and its left and right neighbors (if any) for the case of a row block or its bottom and top neighbors (if any) in the case of a column block are noted. 2) An interchange of the randomly selected cell is performed with either its left (bottom) neighbor and/or its right (top) neighbor for row (column) blocks. For example, in the case of a cell belonging to a row block, if the cell has both left and right neighbors, then one of the neighbors is randomly selected and an interchange of the cell with the selected neighbor is attempted. If the interchange is not accepted, then an interchange is attempted with the neighbor not previously selected. If the cell has only one neighbor, then only that interchange is attempted. 3) An orientation change of the selected cell is attempted if permitted by the user.

The user may also request that TimberWolf is to insert *route-through cells* as necessary if the standard cell circuit contains only row blocks. A route-through cell has two internally connected pins, one on the top and one on the bottom. If a portion of a net must connect two cells which are not on the same row and are not on neighboring rows, then this net must be routed through the rows between those containing the cells. A route-through cell must be inserted to accomplish this for the case of two levels of interconnect. Once the size of the range-limiter window has been reduced such that inter-block cell displacements or interchanges are no longer attempted, TimberWolf will then insert route-through cells as necessary. The route-through cells participate in the generation of new states as described above. That is, they are positioned in their respective rows such that the total wire length objective is minimized.

For standard cell circuits comprised solely of row blocks of cells and pads around the periphery of the blocks, the user may request that TimberWolf is to configure the rows in the most advantegeous manner. The user inputs the number of rows desired and the estimated row separation. For example, in anticipation of the fact that most of the route-through cells are concentrated toward the center-most rows, TimberWolf will restrict the total cell length allowable in these rows. The user supplies an *indent factor* which is the ratio of the total cell length allowed in the center-most row divided by the total cell length allowed in the outermost row. The total cell length allowed in the other rows increases linearly from the center row toward each of the top and bottom rows. TimberWolf also queries the user for the expected number of route-through cells. This can either be a guess or the user may try a short TimberWolf run (that is, with relatively few new states generated at each T) and note the number required. TimberWolf uses this information to increase the actual row lengths. Note that when the final placement is determined by TimberWolf and the route-through cells have been added, the final row lengths will tend to be close to the actual row lengths given to TimberWolf. Having the actual row lengths greater than the total allowable cell length for each row increases the cardinality of the state space of the problem and has been shown to yield the best results.

Of major concern to all implementations of the simulated annealing algorithm is CPU time. The TimberWolf standard cell program was designed to reduce computation time while sacrificing storage. One of the features of the program is that computation time per iteration is constant (that is, it is invariant with the number of cells). The iteration time is defined to be the time required to generate

a new configuration, evaluate the new value of the cost function, and then decide to accept or reject the new configuration. Two key features make this possible. 1) The cells in a block are hashed into bins that partition the block's coordinate system. Hence overlap calculations require a constant amount of time. 2) The possible orientations for a cell, including the pin locations for each orientation, are computed at the outset and are stored. Thus to change a cell orientation, only a pointer change is required rather than recomputing the cell boundaries and pin locations.

Additional reductions in CPU time were achieved by employing a table look-up technique for the computation of the exponential function [6]. This technique requires only 3 table look-ups and 2 floating multiplies to achieve excellent accuracy (it has been observed that the least significant decimal digit is at most plus or minus one in comparison to the exact value of the exponential function). This technique reduced the time per call to the exponential function from 107 to 44 μs on a VAX-780 system and from 75 to 2.5 μs on an IBM-3081/UTS system. Because on the order of several hundred million calls to the exponential function are made for a large standard cell problem, substantial CPU-time reductions were achieved.

Many current standard cell optimization programs attempt to first perform an inter-row optimization and then an intra-row optimization. That is, each cell is first assigned to a row and then in a second step, the cells are placed within their respective row. Note that the method employed by TimberWolf simultaneously considers both optimizations and hence better results should be obtained.

C. Results

The program was interfaced to the CIPAR standard cell placement package developed by American Microsystems, Inc. For the larger circuits tested (800 to 2700 cells), TimberWolf achieved total estimated wire length reductions ranging from 45 to 66 percent in comparison with CIPAR. Furthermore, final chip area reductions ranged from 15 to 57 percent. For a circuit of 1000 cells, TimberWolf reduced the final chip area by 31 percent in comparison to CIPAR and by 21 percent over another commercially available standard cell placement and routing package in a benchmark performed at AMI.

For the largest circuit tested (2700 cells), 75 million iterations were performed. The computation time was 300 μs per iteration (IBM 3081 running UTS), implying nearly 6.5 h of CPU time. TimberWolf runs 12.2 times faster on the IBM/UTS system in comparison to the VAX-780/VMS and VAX-780/UNIX systems.

The memory requirement is linearly related to the number of cells. For the 2700-cell circuit, the memory requirement was 4 Mbytes (32-bit integers are used). The results are summarized in Table I.

The layout of CktA1 using the TimberWolf placement was also compared to the manual layout of the same circuit. A team of designers from AMI worked approximately 4 months on the layout after which time the effort was abandoned for two reasons. First, the projected manual layout was 10-percent larger than the layout produced by CIPAR with TimberWolf, and second, the tape-out deadline had been reached. Manual layouts of circuits CktF and CktG were not attempted by AMI because of the rapid turnaround required by their customer.

TABLE I
TIMBERWOLF STANDARD CELL PLACEMENT OPTIMIZATION PROGRAM

Circuit	# Cells	Total Wire Length Reduction	Final Chip Area Reduction	CPU Time in Hours VAX 780
CktF	2700	66%	57%	84
CktG	1500	**	40%	36
CktA1	1500	45%	30%	20
CktA2	1500	37%	25%	10
CktB	1000	57%	31%	8
CktC	200	41%	15%*	2
CktD	100	37%	15%*	0.5

*pad-limited **not recorded

CktF and CktG were double-metal circuits. Consequently there were many uncommitted route throughs present in each cell. By weighting the vertical net spans approximately one half as much as the horizontal net spans, almost 20-percent additional area reductions were achieved over equal-weighting results.

The CktC and CktD circuits could not have their areas reduced more than 15 percent due to pad limitation. There were two versions of the CktA circuit. The second version had very many of its cells specified to occur in fixed sequences. Hence the number of states in the state space S is significantly reduced. It has been experimentally observed that the wiring area reduction achieved by TimberWolf is less if the cardinality of the state space is reduced.

The effect of the TimberWolf placement optimization can be further demonstrated by the number of route-through cells which were required. For the CktD circuit, the number of route-through cells was reduced from 50 to 14. Furthermore, the number of route throughs was reduced from 51 to zero for the CktC circuit. For the CktB circuit, more than 1000 route-through cells were eliminated. All of the approximately 300 route-through cells were eliminated for the 1500-cell CktA circuit.

The TimberWolf standard cell program was also interfaced to the Zymos placement and routing package (ZYPAR). For a 1000-cell circuit, TimberWolf reduced the total estimated wire length by 44 percent in comparison to ZYPAR. The chip area reduction was limited to 8 percent as a result of using the TimberWolf placement. The smaller-than-expected area reduction was a result of the ZYPAR post-placement row-compaction routine which greatly altered the TimberWolf placement. Modification of the compaction algorithm is under way and much greater area reductions are expected as a result of using TimberWolf.

An interface to TimberWolf was also developed by Intel Corp. Two 1000-cell circuits were used for comparison to their standard cell placement and routing package. The first circuit was manually placed while the second circuit was placed automatically. The result of the TimberWolf placement was a 10-percent final chip area reduction for the first (manually-placed) circuit with a 30-percent reduction in the number of route throughs required. The TimberWolf placement resulted in a 25-percent final chip area reduction for the second circuit.

Furthermore, Hughes Aircraft Company developed an interface to TimberWolf. A 1000-cell circuit was chosen for comparison with their manual placement methodology. The result of the TimberWolf placement was a 6-percent area reduction and a 26-percent reduction in the number of route-through cells that were required for the 1000-cell circuit.

IV. Standard Cell Global Router Program

A. Introduction

The layout of a standard cell circuit often consists of rows of cells bordered by pads and/or buffer circuitry. In order to minimize the need for route-through cells (which increase the area of a circuit), the cells are typically designed with electrically equivalent (internally connected) pins on both the top and bottom side. Thus a net from above can be connected to the top pin while the same net from below can be connected to the bottom pin. The internally connected pins are referred to as a *pin cluster*. A portion of a net which must connect two pin clusters is referred to as a *net segment*.

It often arises that a pin cluster from one cell must be connected to a pin cluster from another cell on the same row. If each such cluster has a top pin and a bottom pin, then this net segment is defined as being *switchable*. A decision must be made as to whether to route the switchable net segment in the channel above or below the row. The TimberWolf global router assigns switchable net segments to channels based on the minimization of the *total channel density*. The total channel density is defined to be the sum of the channel densities for all of the channels.

The TimberWolf global router is applicable to standard cell circuits consisting of rows of cells bordered by pads and/or buffer circuitry. The global router assumes that all necessary route-through cells have been inserted into the proper rows. The global router routes all nets and considers all pins except those nets and pins which route power and ground. It is often the case (as with CIPAR) that separate routines are used to route power and ground. The global router takes into consideration pins on the outer pads or buffer cells.

Some standard cell place and route systems (for example, CIPAR) do not employ a global router. Instead, only a channel router is used and it routes as many connections as possible for each channel. Thus the order in which the channels are routed can have a substantial effect on the total number of wiring tracks required (and thus the area of the circuit). In contrast, after using the TimberWolf global router, specific pins have been identified for interconnection. Thus the number of wiring tracks required is independent of the order in which the channels are routed.

B. Global Router Algorithm

The TimberWolf global router performs the optimization in two stages. The first stage examines each net separately. Two basic steps are applied to each net. 1) The first step identifies which pairs of pin clusters are to be connected based on the minimization of the Manhattan interconnection distance. This results in the identification of the net segments. 2) The second step considers each net segment and selects a pin from each cluster such that the Manhattan length of the segment is minimized. Two pairs of pins are selected for each switchable net segment.

The second stage results in the assignment of a channel for each switchable net segment. The two stages are detailed below.

1) First Stage of the Global Router Algorithm: The first stage consists of applying the two steps detailed below to each net separately.

Step 1

For a given net, the pin clusters that need to be connected are determined. A graph is formed in which the clusters are represented by the nodes and connections between the nodes (the formation of potential net segments) are represented by edges. An edge connects two nodes if a net segment could possibly connect the two clusters. For example, two clusters can be connected only if one of the following two conditions is true. 1) They lie on the same row, with no intervening cluster occupying the same row. This is the case of a potential switchable net segment. The net segment is switchable if each cluster has a pin on the top and on the bottom of the row. That is, the net segment could be routed either in the channel above the row or in the channel below the row. 2) They lie on neighboring rows. Furthermore, there cannot be another cluster lying between the two clusters which occupies either of the rows occupied by the two clusters.

The result of conditions 1) and 2) above is that the maximum degree of a node is 4. Further, this maximum degree is achieved when a given cluster is to be connected to two clusters in the row above (one to the left and one to the right) and to two clusters in the row below (also one to the left and one to the right).

The minimum spanning tree is generated for the graph via Kruskal's algorithm [7]. This portion of the algorithm effectively generates a Steiner tree [8] for the interconnection of the clusters. When the minimum spanning tree has been generated, pairs of pin clusters have been identified which are to be connected by a net segment.

Step 2

In this step, each edge of the minimum spanning tree is examined, and one pin from each cluster is selected to form

the actual net segment. In the case of an edge connecting two clusters on the same row, it is determined if this is a switchable net segment. If the segment is switchable, then two pairs of pins are selected. One pair is for the segment routed in the channel above the row and another pair is for the segment routed in the channel below the row.

Pin selection proceeds as follows. 1) For the case of two clusters on neighboring rows, the bottom pin of the top cluster and the top pin of the bottom cluster are selected based on the minimization of the Manhattan distance between the two points. 2) For the case of two clusters on the same row: a) If the edge is determined to be switchable, the top pin from each cluster is selected based on the minimization of the distance between the two points. Also, the bottom pin from each cluster is similarly selected. b) If the edge is not switchable, either the pair of top pins (if the segment must be routed in the channel above the row) or the pair of bottom pins (if the segment must be routed in the channel below the row) are selected. The pin selection is again based on the minimization of the segment length.

2) Second Stage of the Global Router Algorithm: This step employs a simulated annealing algorithm. The net segments (for all of the nets) with their respective pins are supplied as input. One half of the minimum contact-to-contact spacing is added to each end of the horizontal span of each segment. For each switchable segment, an arbitrary initial selection (of above or below the row) is made. Each channel is examined sequentially to determine its density. The densities of the channels are summed, and this sum is the initial value of the cost function. A new state of the configuration is generated by the random selection of a switchable segment and then routing it on the opposite side of the row from its current position. As a result of the new state, the cost function either increases by 1, decreases by 1, or remains the same. That is, the total channel density changes by at most 1.

The case of no change in the cost is treated further. This is the case in which the net segment switch has no effect on the total channel density. A second *cost function* is introduced in this case. This cost function is a measure of the congestion in a channel between the two points defining the span of a net segment. The cost function is evaluated by taking the difference between the overall channel density and the density between the two points defining the span. The cost function is first evaluated for the span of the net segment in the original channel. Next, the cost function is evaluated for the net segment span in the new channel. The difference in cost (Δc) is determined by subtracting the second cost function value from the first. A negative value of Δc indicates that switching the net segment to the new channel places the segment in a channel of less congestion.

C. Results

The global router was also interfaced to the CIPAR placement and routing package developed by AMI. The global router reduced the number of wiring tracks used by the CIPAR router by 10 to 20 percent. Because routing typically occupies one half of the chip area, this translated to an overall area savings of 6 to 8 percent.

TABLE II
TIMBERWOLF STANDARD CELL OPTIMIZATION PROGRAMS

Circuit	# Cells	Global Router Area Reduction	Final Chip Area Reduction	CPU Time VAX 780 in Hours
CktF	2700	8%	62%	1
CktG	1500	8%	45%	0.5
CktA	1500	6.1%	34%	0.5
CktB	1000	6%	35%	0.3

For the largest circuit (2700-cell CktF), the global router reduced the area by an additional 8 percent. A total area savings of 62 percent was achieved for CktF when both TimberWolf placement optimization and the global router were applied. The results are summarized in Table II, showing the additional area reductions due to use of the global router and also the overall area reductions as a result of using both TimberWolf placement and global routing.

Simulation results for CktG revealed that all interconnections had capacitance values below the specifications, and hence that the circuit should operate properly at the specific clock rate. Simulation results for CktF were not available at this time.

Fig. 2 depicts the layout of a 1500 cell circuit which was produced by CIPAR. The layout as a result of using TimberWolf for placement and global routing is shown in Fig. 3. Note that the TimberWolf layout was pad limited and hence the area reduction achieved was limited to 11 percent. However, the core size (the area inside the pad ring) was reduced by 22 percent in area.

V. MACRO/CUSTOM PLACEMENT OPTIMIZATION PROGRAM

A. Introduction

This program optimizes the placement of macro cells and custom cells, as well as pads. The term *macro cell* will be used to refer to a cell contained in a cell library. That is, the dimensions of the cell are known, as are the pin locations. The term *custom cell* will be used to refer to a block of circuitry known only to occupy an estimated area and to possess a list of pins.

The program places circuits comprised solely of macro cells as well as circuits comprised entirely of custom cells. Furthermore, the program will place circuits consisting of a combination of macro and custom cells. The macro cells and custom cells may be of any rectilinear shape.

TimberWolf allows the specification of lower and upper bounds for the aspect ratio of a custom cell. If a range of aspect ratios is given for a custom cell, TimberWolf will try to select the shape of the cell which minimizes chip area.

Wire length calculations are based on the exact pin locations. Thus all possible orientations are considered for each cell.

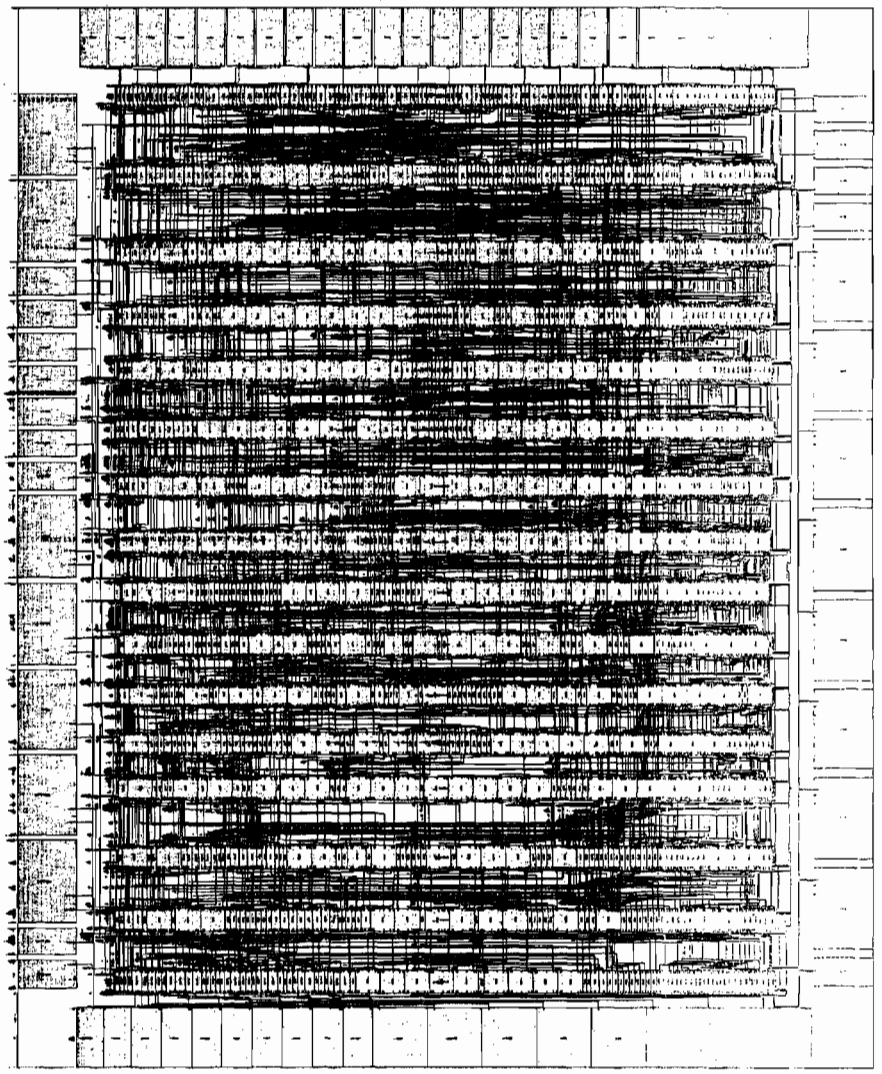


Fig. 2. CIPAR layout of 1500-cell circuit without TimberWolf.

Another feature of TimberWolf is the multiple region capability. This feature incorporates either a division of the chip into regions or the placement of multiple chips simultaneously. Interchanges of cells from different regions are permitted only if the regions belong to the same exchange class. The exchange class mechanism is extended to individual cells as well.

Pins are specified in several possible ways. 1) A pin may be given a particular fixed location. 2) A pin may be assigned to a particular side or sides of the cell. 3) A group of pins may be assigned to a particular side or sides of a cell. 4) A group of pins may be assigned to a particular sequence as well as a particular side or sides.

B. Macro/Custom Cell Placement Algorithm

For macro and custom cells, there are often pins on all of the sides of the cells. Consequently, wiring space must be allocated around each cell. If insufficient space is allocated during TimberWolf placement, the global and detailed routers will have to (perhaps substantially) alter the placement. The strategy employed by TimberWolf to ensure routability with a minimum amount of placement alteration during routing consists of the following: TimberWolf (by default) computes the expected wiring area required along each side of each cell based on the number of pins on that side. Appropriate borders are then appended around the enclosed area of the cell. This prevents cells from abutting in the final placement and hence allows approximately sufficient wiring space around each cell. Furthermore, TimberWolf allows the user to override the default border values.

The number of possible locations at which an *uncommitted* pin could be placed on a custom cell can often number into the thousands. Execution time considerations (as in the standard cell program) require that the pin locations be stored for each orientation of the cell. Clearly the amount of storage required can become excessively large. This potential problem is averted by defining a

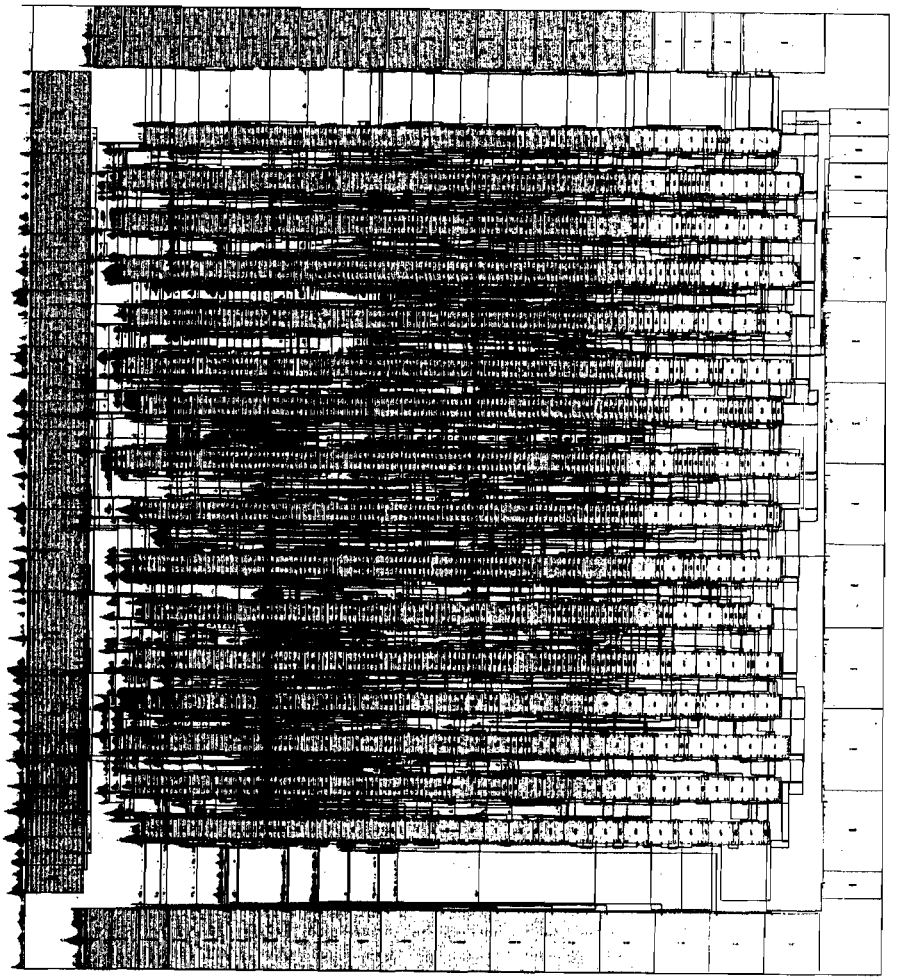


Fig. 3. CIPAR layout of 1500-cell circuit with TimberWolf placement and global routing.

specified number of pin *sites* approximately evenly spaced along the periphery of a cell. Furthermore, each site is assigned a *capacity*. The capacity is a function of the number of pin locations encompassed by the site. During the annealing stages, pins are assigned to sites. Upon completion of the annealing algorithm, the pins for a given site are assigned to locations within the scope of the site based on the minimization of wire length. For accuracy considerations, the number of pin sites that are declared for a given placement problem is usually limited only by memory capacity.

The location of the pins on a macro cell are taken exactly. That is, their location is not approximated by the pin-site mechanism. The same is true for fixed-location pins on custom cells (if any are so specified). The capacity for a site in the vicinity of a fixed-location pin is correspondingly reduced.

The cost function consists of two independent parts. The first part is the total estimated wire length which is based on the sum over all nets of the half-perimeter of a net's bounding box. The second is the penalty function. The penalty function consists of two parts. 1) The first part is the sum of the overlap penalties for the cells. This penalty function was incorporated because of the usual difference in the size and shape of the cells. Often two cells are selected for interchange which differ in size and/or shape. Therefore, an exchange of location of these two cells often results in some overlap with one or more of the cells. Furthermore, the program often selects a single cell for a displacement to a new location or an aspect ratio change (in the case of custom cells). Once again, some overlap may result. The penalty assessed for an overlap of two cells is equal to the square of the quantity of the area of overlap (including cell borders) plus an offset value. The offset parameter is selected to ensure that as the parameter T approaches zero, then the total overlap approaches zero. 2) The second part is the sum of the penalties assessed for the contents of a pin site exceeding its capacity. When a pin is displaced from an original site to a new site, the contents of the old site is reduced by 1 and the contents of the new site is increased by 1. The penalty assessed for a site is a product of the square of the amount by which the contents exceed the capacity, times a factor inversely related to the capacity of the site. This factor reflects the fact that exceeding the capacity by a given amount is a more serious violation for the sites with smaller capacities.

New states can be generated in several possible ways. 1) A pair of cells (either could be a macro cell or a custom cell) are selected for interchange. 2) A single cell is selected for a displacement to a new location. 3) A single cell is selected for an orientation change. 4) A custom cell is selected for an aspect ratio change. 5) An uncommitted pin (or sequence of pins) is assigned to a new site (or sites).

The ratio of single cell displacements to cell interchanges has a significant effect on the quality of the final placement. Initial experimental investigation has revealed that the best results are obtained when the ratio is about 10 to 1.

The strategy for generating new states is based on the following: 1) A random number between one and the number of cells is generated. The cells are numbered sequentially from one. 2) A second random number is generated between 1 and the number of cells times 10. 3) If the two numbers both represent cells, then the pair of cells are interchanged to generate a new state. 4) If only the first number represents a cell, then the new state is generated by the displacement of the cell to a randomly selected location. If this new state was rejected, the next state generated is an orientation change for the cell. Similarly, if this new state was rejected and if the cell is a custom cell, then the next state is an aspect ratio change. Finally, if this new state was rejected, then a new state is generated by the selection of an uncommitted pin or group of uncommitted pins for transfer to a new pin site or sites.

As in the case of standard cell placement, the displacement of a cell to a new location is controlled by a *range limiter*, which limits the range of the displacement of a cell. For example, in the latter stages of the algorithm when the value of T approaches zero, the displacement of a cell has very little chance of being accepted unless the displacement is very local. By limiting the range of the cell displacements in the latter stages of the algorithm, the cells undergo many small displacements while gradually eliminating overlaps and reducing wire length.

The implementation of the range limiter is as follows. A rectangular *window* is centered at the center of the cell to be displaced and this window has a particular horizontal span and a particular vertical span. At the beginning of the algorithm, when T is at its maximum value, the horizontal span of the window is equal to twice the horizontal span of the chip and similarly the vertical span of the window is equal to twice the vertical span of the chip. The horizontal and vertical window spans are proportional to the logarithm of the value of T. Hence, when the value of T is reduced, the size of the window is correspondingly reduced. When a cell is to be displaced, a randomly selected location within the window is chosen as the new location for the cell. That is, a region is randomly selected which intersects the window and then a random position is selected within that region and within the window.

Pairwise interchanges of cells are also controlled by the range limiter. An interchange of two cells is attempted only if the window can be positioned such that it contains the centers of both cells.

This program is also applicable to printed circuit board placement problems. The circuits to be placed are handled in the same manner as macro cells, that is, cells with fixed geometry and fixed pin locations. In printed circuit board layouts, total wire length and maximum wire length minimization are important objectives per se in addition to their correlation with ease of routing. In fact, signal crosstalk due to long wires may cause signal degradation and limit the speed of operation much more severely than in integrated circuits.

C. Results

The TimberWolf macro/custom cell placement optimization program is currently being interfaced to CIPAR for testing purposes. In addition, testing is in progress on some macro cell circuits designed at UC Berkeley.

This program was applied to a Honeywell Information Systems Italy printed circuit board problem in which the circuits had variable size. The processor board required the placement of 613 circuits, each of which had from 2 to 64 pins. The circuits had to be placed on a 14.4×16 in printed circuit board. The processor board had 900 nets, 4000 pins, and contained 3 microprocessors.

TimberWolf used 18 h of CPU time on a VAX-780/UNIX system to place the circuits. The placement obtained was routed by the HONDA (Honeywell Design Automation) printed circuit router and 96-percent routing completion was achieved.

For comparison, the manual placement of the same printed circuit board was considered. The total estimated wire length of the TimberWolf placement was 21-percent less than the manual placement. HONDA was also run on the manual placement resulting in 99-percent routing completion. The TimberWolf placement resulted in a 10-percent reduction in actual total wire length. Furthermore, the manual placement required approximately 4 months of effort on the part of the design team.

These results are preliminary since a few constraints deriving from the automatic insertion of components on the printed circuit board were neglected by TimberWolf. Furthermore, the HONDA router is specifically tuned to a particular layout style, and hence is not fully compatible with an automatic layout program such as TimberWolf. Minor modifications to the router should produce improvements in the final results.

VI. Gate-Array Placement Optimization Program

A. Introduction

This section describes the generalized gate-array placement program. Each fundamental unit in a gate array will be referred to as a *cell*. Hence, a 50 by 50 gate array is said to have 2500 cells. Some gate array designs allow additional flexibility and hence greater gate utilization by creating functionally independent units within a cell. For

example, Tektronix gate arrays widely utilize functional units which are half-cell sized. TimberWolf allows the functional units to be half-cell sized or quarter-cell sized. The term *module* will refer to a fundamental unit specified in the net list. A module may be the size of: 1) a full cell, 2) a half cell, or 3) a quarter cell. Additionally, macro modules may be specified. A macro module consists of a prewired, arbitrarily shaped collection of cells.

TimberWolf has other features which provide additional flexibility. For example, a module (or macro module) may be designated as unmoveable (that is, preplaced) or as belonging to an exchange class of modules. The modules in such a class may only be interchanged among themselves. This feature is often desirable when a group of modules on the edge of the gate array are to be considered as primary terminals. Often the exact location of a given primary terminal is not important, only that it lie on a given edge.

It is often the case that gate arrays have wider channels in the center of the array. This is in anticipation of the greatest wiring congestion occurring in this region. Because prewired macro modules usually have a fixed cell-to-cell spacing, certain macros may not be placed in the center region (or the outer regions). TimberWolf allows the designation of cell locations as either suitable or unsuitable for a particular set of macro modules.

B. Gate-Array Placement Algorithm

The TimberWolf gate array placement program can be used with either of two cost functions. The first cost function is based on the computation of net crossing *histograms* for each horizontal and vertical channel of the placement region. The histograms are computed by considering the bounding box of each net and adding 1 to the histogram for each channel intersecting the bounding box. The sum of the histogram values for each horizontal and vertical channel is equivalent to summing the half perimeters of the bounding boxes of each net. Further, a net-crossing *threshold* value is assigned to each channel. If the number of nets crossing a channel exceeds the specified threshold value, a penalty is assessed proportional to the square of the number of net crossings exceeding the threshold. The threshold mechanism has the effect of evening out the wiring congestion during the earlier stages of the annealing. This has shown to result in a lower value of the total wire length. A partitioning effect may be produced by setting the threshold of a particular channel to zero or a negative value. In this case, nets crossing this channel will be severely penalized. The formulation of the cost function in terms of net-crossing histograms and threshold values was first introduced by Kirkpatrick, Gelatt, and Vecchi [1].

A second cost function for this program examines the local routing congestion more closely. For this cost function, each *channel segment* is assigned a threshold value. A channel segment is a portion of a horizontal or vertical channel with a length equal to the cell-center to cell-center spacing in that region of the array. For example, if the bounding box of a net encompasses 2 cells in the horizontal direction and 3 cells in the vertical direction, then a total of 17 segments are enclosed by the bounding box. The congestion per channel segment introduced by this net is approximated as the half perimeter of the bounding box 5 divided by the total number of segments enclosed 17. The factor of 5/17 is the estimated probability of occupancy for the given net in each of the 17 segments. The given net contributes zero to all other segments. The summation of the occupancy probabilities over all nets for a given segment is an estimate of the number of wiring tracks required. The cost function is then the sum of the expected occupancy of each segment plus a penalty assessed for each segment which has occupancy exceeding the corresponding threshold. Specifying a threshold value for each channel segment which reflects the actual fixed channel width increases the likelihood that the final placement will be routable. Furthermore, the total wire length will be minimized within the limits of these constraints.

TABLE III
TIMBERWOLF GATE ARRAY PLACEMENT PROGRAM

Circuit (# modules)	Stevens	Goto and Kuh	TimberWolf	CPU Time in Mins.
151	2181	2098	1731	15
108	untested	1242	909	10
67	700	618	580	5

C. Results

Experiments are currently being initiated on large gate array problems. To test the program and compare it with existing placement techniques, a set of standard benchmarks have been considered. These benchmarks are the ILLIAC IV computer boards reported by Stevens [9]. Note that the printed circuit board problem as stated for these examples is a particular case of the general gate array placement problem described in the previous subsection.

Wire length for a net was estimated by computing one half of the perimeter of the net's bounding box. The figure of merit is the sum of the estimated wire lengths for each net.

Three of the ILLIAC IV computer boards were tested. 1) The largest example required the placement of 151 modules on an 11×15 board. TimberWolf reduced the total wire length by 21 percent over Stevens' result and by 17 percent over the result published by Goto and Kuh [10]. 2) The second example required the placement of 108 modules on an 8×15 board. TimberWolf reduced the total wire length by 27 percent over the result published by Goto and Kuh. 3) The third example required the placement of 67 modules on a 5×15 board. TimberWolf reduced the total wire length by 17 percent over Stevens' result and 6 percent over the result published by Goto and Kuh.

The value of α remained at a constant value of 0.90 for each of the examples. The results are summarized in Table III. CPU times are for a VAX 11/780 running UNIX.

VII. Conclusions

The TimberWolf placement and routing package has been shown to provide substantial chip area savings in comparison to existing standard cell layout programs. Substantial wire length reductions were also achieved for the gate array placement program for some benchmark examples. The TimberWolf macro/custom program is applicable to placement problems as complex as a multichip design employing a combination of macro cells and custom cells. The macro/custom program was applied to an industrial circuit board problem and improved the manual placement by 10 percent in terms of total (exact) wire length.

The TimberWolf placement and routing package is written in the C programming language. The package currently runs under both the VAX/UNIX and VAX/VMS operating systems as well as the IBM/UTS system. The package is easily convertible to other systems supporting the C language.

Acknowledgment

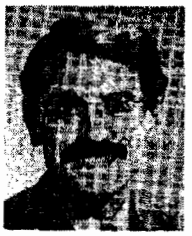
The authors would like to thank American Microsystems, Inc. for allowing the interface of TimberWolf to the CIPAR system and for providing the test circuits for the standard cell package. The authors would also like to thank Honeywell Information Systems Italy for providing the test case for the printed circuit board package. Special thanks are also extended to Intel Corp. for providing computer time on an IBM 3081 for TimberWolf testing.

The support of J. Tobias and B. Kirk of AMI, S. Nachtsheim of Intel, and D. Cesa Bianchi, L. Fezzi and M. Vinsani of HISI is gratefully acknowledged. Further, the authors deeply appreciate the efforts of T. Young of Zymos and C. P. Hsu of Hughes Aircraft Co. in developing TimberWolf interfaces and for providing standard cell test circuits.

The authors wish to thank F. Romeo and K. Keller for stimulating discussions. C. Sechen wishes to thank P. Moore, T. Quarles, R. Spickelmier, and M. Hofmann for their significant contributions to his knowledge of the C programming language and the UNIX operating system.

References

[1] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by simulated annealing," IBM Computer Science/Engineering Technology Watson Res. Center, Yorktown Heights, NY, Tech. Rep., 1982.
[2] M. Vecchi and S. Kirkpatrick, Global Wiring by Simulated Annealing, *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 215–222, 1983.
[3] F. Romeo and A. Sangiovanni-Vincentelli, Probabilistic Hill Climbing Algorithms: Properties and Applications, ERL, College of Engineering, Univ. California, Berkeley, CA, Mar. 1984.
[4] D. Geman and S. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," unpublished.
[5] M. Lundy and A. Mees, "Convergence of the annealing algorithm," unpublished.
[6] J. Deutsch, private communication, UC-Berkeley, 1984.
[7] J. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. Amer. Math Soc.*, vol. 7, no. 1 pp. 48–50, 1956.
[8] M. Hanan, "Net wiring for large scale integrated circuits," IBM Res. Rep. RC-1375, IBM, Feb. 1965.
[9] J. Stevens, "Fast heuristic techniques for placing and wiring printed circuit boards," Ph.D dissertation, Univ. of Illinois, Urbana, IL, 1972.
[10] S. Goto and E. Kuh, "An approach to the two-dimensional placement problem in circuit layout," *IEEE Trans. Circuits Syst.*, vol. 25, p. 208,. Apr. 1978.



Alberto Sangiovanni-Vincentelli (M'74–SM'81–F'83) received the Dr. Eng. degree (summa cum Laude) from the Politecnico di Milano, Italy, in 1971.

From 1971 to 1977, he was with the Istituto di Elettrotecnica ed Elettronica, Politecnico di Milano, Italy. In 1976 he joined the Department of Electrical Engineering and Computer Sciences of the University of California at Berkeley, where he is presently Professor and Vice Chairman. He is a Consultant in the area of computer-aided design to several industries including IBM, AT&T Bell Laboratories, Harris Semiconductors, and General Electric. His research interests are in various aspects of computer-aided design of integrated circuits, with particular emphasis on VLSI simulation and optimization. He was Associate Editor of the IEEE Transactions on Circuits and Systems, Associate Editor of the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. He is an Associate Editor of the IEEE Design and Test Magazine, a member of the Large-Scale Systems Committee of the IEEE Circuits and Systems Society and of the Computer-Aided Network Design (CANDE) Committee. He was the Guest Editor of a Special Issue of the IEEE Transactions on Circuits and Systems on CAD for VLSI. He has been elected Executive Vice-President of the IEEE Circuits and Systems Society in 1983 in charge of Membership Development. He is also Division I Representative in the IEEE Membership Development Committee.

In 1981 he received the Distinguished Teaching Award of the University of California. At the 1982 IEEE-ACM Design Automation Conference he was given a Best Paper and a Best Presentation Award. In 1983 he received the Guillemin–Cauer Award for the best paper published in the Transactions on CAS and CAD in 1981–1982. At the 1983 Design Automation Conference he received a Best Paper Award.

Dr. Sangiovanni-Vincentelli is member of ACM and Eta Kappa Nu.

Carl Sechen received the B.E.E. degree (with highest honors) from the University of Minnesota in 1975. He received the M.S. degree from M.I.T. in 1977. In 1977 and 1978 he was with Honeywell Systems and Research Center. Since 1979 he has been a graduate student at the University of California at Berkeley where he is completing the requirements for the Ph.D. degree.

His research interests are in many aspects of computer-aided design of VLSI circuits, with emphasis on automated layout. He is working at several companies including Intel, American Microsystems, Inc., Hughes Aircraft Co., Harris Semiconductors, and Zymos Corp.

Mr. Sechen is a member of Eta Kappa Nu, Tau Beta Pi, and Sigma Xi.